Gpipe Documentation

Release 0.6.2

Andreas Heger

Nov 24, 2021

CONTENTS

1 Mission statement **3**
- 1.1 Other toolkits with similar functionality . . . 3

2 Installation instructions **5**
- 2.1 Quick installation . . . 5

3 Using CGAT Tools **7**
- 3.1 Command line usage . . . 7
- 3.2 Indexing genomes . . . 9
- 3.3 Pipeline usage . . . 9

4 Tool map **11**

5 Tool reference **13**
- 5.1 Genomic intervals/features . . . 13
- 5.2 Gene sets . . . 14
- 5.3 Sequence data . . . 14
- 5.4 NGS data . . . 14
- 5.5 Variants . . . 15
- 5.6 Genomics . . . 15

6 Contributing to CGAT code **17**
- 6.1 Checklist for new scripts/modules . . . 17
- 6.2 Building extensions . . . 17
- 6.3 Writing recipes . . . 18
- 6.4 Writing pipelines . . . 19

7 Reference **21**
- 7.1 Repository layout . . . 21
- 7.2 API . . . 21

8 Release Notes **239**
- 8.1 Release 0.4.0 . . . 239

9 Quickstart **241**

10 Developer's guide **243**
- 10.1 Testing . . . 243
- 10.2 Style Guide . . . 246
- 10.3 Documentation . . . 252
- 10.4 Importing CGAT scripts into galaxy . . . 254

11 Glossary 257

12 Disclaimer 259

Python Module Index 261

Index 263

CGAT is a collection of tools for the computational genomicist written in the Python language (Should work with Python 2.7, but we only actively support Python 3.6+). The tools have been developed and accumulated in various genome projects (Heger & Ponting, 2007, Warren et al., 2008) and NGS projects (Ramagopalan et al., 2010). The tools are in continuous development. The tools work from the command line, but can readily be installed within frameworks such as Galaxy.

The documentation below covers the script published in Bioinformatics.

Detailed instructions on installation, on usage and a tool reference are below, followed by a *Quickstart* guide.

CHAPTER

ONE

MISSION STATEMENT

The CGAT-apps code collection has been written over several years in the context of comparative genomics and more recently next-generation sequencing analysis.

The aim of the toolkit is to solve practical problems in the analysis of genomic data. The focus of the toolkit is to facilitate the interpretation of genomic data in a biological context. Furthermore, as a training institution our aim is to write code that is well structured and can serve as an introduction to advanced bioinformatic scripting for biologists.

1.1 Other toolkits with similar functionality

The CGAT code collection extends, complements but also overlaps various other toolkits. As all toolkits, and ours, continue to evolve, this is a very dynamic relationship. For example, our workflows frequently use other toolkits, in particular bedtools and the UCSC tools, for high-performance computations. Usage of common genomic file formats and a command line interface ensures compatibility.

Below is a list of toolkits with similar or complementarity functionality to the CGAT code collection and quotes from their respective web-sites:

- bedtools The BEDTools utilities allow one to address common genomics tasks such as finding feature overlaps and computing coverage. The utilities are largely based on four widely-used file formats: BED, GFF/GTF, VCF, and SAM/BAM. Using BEDTools, one can develop sophisticated pipelines that answer complicated research questions by "streaming" several BEDTools together.
- samtools SAM Tools provide various utilities for manipulating alignments in the SAM format, including sorting, merging, indexing and generating alignments in a per-position format.
- UCSC tools Jim Kent's genomic utilities for working with genomic features and alignments.
- EMBOSS EMBOSS is "The European Molecular Biology Open Software Suite". EMBOSS is a free Open Source software analysis package specially developed for the needs of the molecular biology (e.g. EMBnet) user community. The software automatically copes with data in a variety of formats and even allows transparent retrieval of sequence data from the web. Also, as extensive libraries are provided with the package, it is a platform to allow other scientists to develop and release software in true open source spirit. EMBOSS also integrates a range of currently available packages and tools for sequence analysis into a seamless whole. EMBOSS breaks the historical trend towards commercial software packages.
- GROK GROK (Genomic Region Operation Toolkit) is "Swiss Army knife" library for processing genomic interval data. GROK operates on genomic regions, annotated chromosomal intervals that represent sequencing short reads, gene locations, ChIP-seq peaks or other genomic features. Applications of GROK include file format conversions, set operations, overlap queries, and filtering and transformation operations. Supported file formats include BAM/SAM, BED, BedGraph, CSV, FASTQ, GFF/GTF, VCF and Wiggle.
- biopieces The Biopieces are a collection of bioinformatics tools that can be pieced together in a very easy and flexible manner to perform both simple and complex tasks. The Biopieces work on a data stream in such a

way that the data stream can be passed through several different Biopieces, each performing one specific task: modifying or adding records to the data stream, creating plots, or uploading data to databases and web services.

- fastx-toolkit The FASTX-Toolkit is a collection of command line tools for Short-Reads FASTA/FASTQ files preprocessing.

CHAPTER

TWO

INSTALLATION INSTRUCTIONS

The section below describes how to install the cgat-apps. We distinguish between two different installation types: production and development. The former refers to the released version of our tools, which can be installed using pip or conda, and is the recommended installation for users. The latter refers to the installation of the development version of the apps, which can be used to make changes to our code base.

Please note that we can not test our code on all systems and configurations out there so please bear with us.

2.1 Quick installation

2.1.1 Install using Conda

2.1.2 Conda Installation

The preferred method to install CGAT Apps is using the installation script, which uses mamba, the fast C implementation of conda.

To install cgat-apps using mamba:

```
mamba install -c conda-forge -c bioconda cgat-apps
```

2.1.3 Developers: try the installation script

Here are the steps:

```
# Install conda and mamba according the the documentation. Next
# install the conda packages for cgat-apps to work
conda env create -f conda/environments/cgat-apps.yml

# enable the conda environment
conda activate cgat-a

# Install the development version of cgat-apps
python setup.py develop

# finally, please run the cgat command-line tool to check the installation:
cgat --help
```

The installation script will put everything under the specified location. The aim is to provide a portable installation that does not interfere with the existing software. As a result, you will get a conda environment working with CGAT Apps which can be enabled on demand according to your needs.

2.1.4 Install using pip

You can also use pip to install the CGAT scripts. To go down this route, please type:

```
pip install cgat
```

However, cgat-apps depends on numerous other python packages which themselves might require manual intervention.

CHAPTER

THREE

USING CGAT TOOLS

3.1 Command line usage

CGAT tools are written for command line usage with a consistent interface that makes them amenable to integration in pipelines. Tools can be accessed through the `cgat` front-end that will be installed in your PATH.

To get a list of all available commands, type:

```
cgat --help
```

Command line help for individual tools is available through each tool's `--help` option:

```
cgat gff2gff --help
```

3.1.1 Logging

CGAT scripts output logging information as comments starting with a # into stdout or into a separate log file (`--log`).

Below is an example of logging output:

```
# output generated by /ifs/devel/andreas/cgat/beds2beds.py --force-output --exclusive-
↪overlap --method=unmerged-combinations --output-filename-pattern=030m.intersection.
↪tsv.dir/030m.intersection.tsv-%s.bed.gz --log=030m.intersection.tsv.log Irf5-030m-
↪R1.bed.gz Rela-030m-R1.bed.gz
# job started at Thu Mar 29 13:06:33 2012 on cgat150.anat.ox.ac.uk -- e1c16e80-03a1-
↪4023-9417-f3e44e33bdcd
# pid: 16649, system: Linux 2.6.32-220.7.1.el6.x86_64 #1 SMP Fri Feb 10 15:22:22 EST␣
↪2012 x86_64
# exclusive                             : True
# filename_update                       : None
# ignore_strand                         : False
# loglevel                              : 1
# method                                : unmerged-combinations
# output_filename_pattern               : 030m.intersection.tsv.dir/030m.
↪intersection.tsv-%s.bed.gz
# output_force                          : True
# pattern_id                            : (.*).bed.gz
# stderr                                : <open file \'<stderr>\', mode \'w\' at␣
↪0x2ba70e0c2270>
# stdin                                 : <open file \'<stdin>\', mode \'r\' at␣
↪0x2ba70e0c2150>
# stdlog                                : <open file \'030m.intersection.tsv.log\',␣
↪mode \'a\' at 0x1f1a810>
```

(continues on next page)

(continued from previous page)

```
# stdout                                       : <open file \'<stdout>\', mode \'w\' at␣
↪0x2ba70e0c21e0>
# timeit_file                                  : None
# timeit_header                                : None
# timeit_name                                  : all
# tracks                                       : None
```

The header contains information about:

- the script name (`beds2beds.py`)
- the command line options (`--force-output --exclusive-overlap --method=unmerged-combinations --output-filename-pattern=030m.intersection.tsv.dir/030m.intersection.tsv-%s.bed.gz --log=030m.intersection.tsv.log Irf5-030m-R1.bed.gz Rela-030m-R1.bed.gz`)
- the time when the job was started (`Thu Mar 29 13:06:33 2012`)
- the location it was executed (`cgat150.anat.ox.ac.uk`)
- a unique job id (`e1c16e80-03a1-4023-9417-f3e44e33bdcd`)
- the pid of the job (`16649`)
- the system specification (`Linux 2.6.32-220.7.1.el6.x86_64 #1 SMP Fri Feb 10 15:22:22 EST 2012 x86_64`)

Once completed successfully, a script will output to the logfile. Below is typical output:

```
# job finished in 11 seconds at Thu Mar 29 13:06:44 2012 -- 11.36  0.45  0.00  0.01 --
↪ e1c16e80-03a1-4023-9417-f3e44e33bdcd
```

The footer contains information about:

- the job has finished (`job finished`)
- the time it took to execute (`11 seconds`)
- when it completed (`Thu Mar 29 13:06:44 2012`)
- **some benchmarking information (`11.36 0.45 0.00 0.01`) which is** `user time`, `system time`, `child user time`, `child system time`.
- the unique job id (`e1c16e80-03a1-4023-9417-f3e44e33bdcd`)

The unique job id can be used to easily retrieve matching information from a concatenation of log files.

The logging level can be determined by the `--verbose` option. A level of `0` means no logging output, while `1` is information messages only, while `2` outputs also debugging information.

3.1.2 I/O redirection

Most scripts work by reading data from *stdin* and outputting data to *stdout*. Both can be redirected to files with the `-I/--stdin` and `-O/--stdout` options. *stderr* can be redirected with `-E/--stderr`.

3.2 Indexing genomes

Many CGAT tools require genomic information, some require the actual genomic sequence, but many require information about chromosome sizes. Thus, many tools have the obligatory option `--genome-file`.

The `genome-file` argument points to an indexed fasta file. CGAT tools can read two different indices, either files indexed using the supplied *index_fasta.py - Index fasta formatted files* script or using the samtools `faidx` command.

3.3 Pipeline usage

We use a light-weight workflow system called ruffus, but others are equally possible such as galaxy (see GalaxyInstallation). These tools allow CGAT tools to run in an automated fashion.

Using unix pipes, CGAT tools can also be easily run in a parallel fashion. For example, we have a script called *farm.py* (not part of the CGAT collection, but within the CGAT repository), that allows to split input data and run separate chunks on our compute cluster. Below is a simple example of running the command:

```
zcat geneset.gtf.gz
| cgat gtf2table --counter=length --log=log |
gzip > out.tsv.gz
```

in parallel on the cluster, running one job per chromosome:

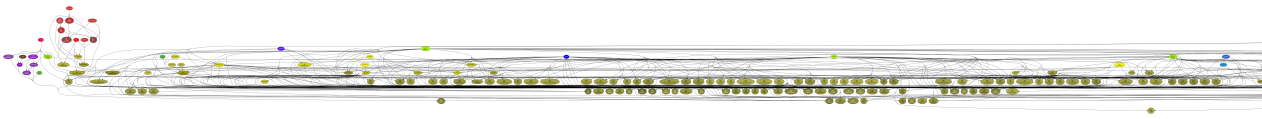
```
zcat geneset.gtf.gz
| farm.py --split-at-column=1
        "cgat gtf2table --counter=length --log=log"
| gzip
> out.tsv.gz
```

CHAPTER

FOUR

TOOL MAP

This page contains a graph visualization of some of the tools within the cgat-apps collection.

Click to view the 

CHAPTER

FIVE

TOOL REFERENCE

This page summarizes prominent tools within the CGAT Code collection. The tools are grouped losely by functionality.

5.1 Genomic intervals/features

beds2counts - compute overlap stats between multiple bed files Compute overlap statistics of multiple *bed* files.

bed2fasta.py - get sequences from bed file Transform interval data in a *bed* formatted file into a *fasta* formatted file of sequence data.

bed2gff.py - convert bed to gff/gtf Convert between interval data. Convert a *bed* formatted file to a *gff* or *gtf* formatted file.

gff2gff.py - manipulate gff files Work on *gff* formatted files with genomic features. This tools sorts/renames feature files, reconciles chromosome names, and more.

bed2bed - manipulate bed files Filter or merge interval data in a *bed* formatted file.

bed2graph.py - compute the overlap graph between two bed files Compare two sets of genomic intervals and output a list of overlapping features.

bed2stats.py - summary of bed file contents Compute summary statistics of genomic intervals.

<no title> Annotate genomic intervals (composition, peak location, overlap, . . .)

beds2beds.py - decompose bed files Decompose multiple sets of genomic intervals into various intersections and unions.

diff_bed.py - count differences between several bed files Compare multiple sets of interval data sets. The tools computes all-vs-all pairwise overlap summaries. Permits incremental updates of similarity table.

gff2bed.py - convert from gff/gtf to bed Convert between formats

split_gff - split a gff file into chunks Split a file in gff format into smaller files. The script ensures that overlapping intervals remain in the same file.

gff2coverage.py - compute genomic coverage of gff intervals This script computes the genomic coverage of intervals in a *gff* formatted file. The coverage is computed per feature.

gff2fasta.py - output sequences from genomic features Output genomic sequences from intervals.

gff2histogram.py - compute histograms from intervals in gff or bed format Compute distributions of interval sizes, intersegmental distances and interval ovelap from list of intervals.

gff2stats.py - count features, etc. in gff file Summarize features within a *gff* formatted file.

gff2psl.py - convert from gff to psl Convert between formats.

5.2 Gene sets

gtf2gff.py - convert a transcript set to genomic features Translate a gene set into genomic annotations such as introns, intergenic regions, regulatory domains, etc.

<no title> Annotate transcripts in a *gtf* formatted file. Annotations can be in reference to a second gene set (fragments, extensions), aligned reads (coverage, intron overrun, . . .) or densities.

gtf2fasta.py - annotate genomic bases from a gene set Annotate each base in the genome according to its use within a transcript. Outputs lists of junctions.

gtf2gtf.py - manipulate transcript models merge exons/transcripts/genes, filter transcripts/genes, rename transcripts/genes, . . .

gtf2tsv.py - convert gtf file to a tab-separated table convert gene set in *gtf* format to tabular format.

gtfs2tsv.py - compare two genesets Compare two gene sets - output common and unique lists of genes.

diff_gtf.py - compute overlap between multiple gtf files Compare multiple gene sets. The tools computes all-vs-all pairwise overlap of exons, bases and genes. Permits incremental updates of similarity table.

5.3 Sequence data

fastqs2fasta.py - interleave two fastq files Interleave paired reads from two fastq files into a single fasta file.

index_fasta.py - Index fasta formatted files Build an index for a fasta file. Pre-requisite for many CGAT tools.

fasta2kmercontent.py Count kmer content in a set of *fasta* sequences.

<no title> Compute features of sequences in *fasta* formatted files

diff_fasta.py - compare contents of two fasta files Compare two sets of sequences. Outputs missing, identical and fragmented sequences.

fasta2bed.py - segment sequences Segment sequences based on G+C content, gaps, . . .

fastas2fasta.py - concatenate sequences from multiple fasta files Concatentate sequences from multiple files.

fasta2variants.py - create sequence variants from a set of sequences In-silico creation of variants of protein coding sequences.

5.4 NGS data

bam2geneprofile.py - build meta-gene profile for a set of transcripts/genes Compute meta-gene profiles from aligned reads in a *bam* formatted file. Also accepts *bed* or *bigwig* formatted files.

bam2bam.py - modify bam files Operate on *bam* formatted files - filtering, stripping, setting flags.

bam2bed.py - convert bam formatted file to bed formatted file Convert *bam* formatted file of genomic alignments into genomic intervals. Permits merging of paired read data and filtering by insert-size.

bam2fastq.py - output fastq files from a bam-file Save sequence and quality information from a *bam* formatted file.

bam2peakshape.py - compute peak shape features from a bam-file Compute read densities over a collection of intervals. Also accepts *bed* or *bigwig* formatted files.

Purpose Compute summary statistics of a *bam* formatted file.

bam2wiggle.py - convert bam to wig/bigwig file Convert read coverage in a *bam* formatted file into a *wiggle* or *bigwig* formatted file.

bam_vs_gtf.py - compare bam file against gene set Compute stats on exon over-/underrun and spliced reads.

bam_vs_bed.py - count context that reads map to Compute coverage of reads within multiple interval types.

bam_vs_bam.py - compute coverage correlation between bam files Outputs side-by-side comparison of residue level counts between multiple *bam* formatted files.

fastq2fastq.py - manipulate fastq files Perform quality score conversion between *fastq* formatted files.

fastqs2fasta.py - interleave two fastq files Interleave paired end data.

fastq2table.py - compute stats on reads in fastq files Output bases below quality threshold, number of N's, quality score distribution.

fastqs2fastqs.py - manipulate (merge/reconcile) fastq files Ensure that paired read *fastq* formatted files are consistent after filtering on the individual files.

diff_bam.py - compare multiple bam files against each other Perform read-by-read comparison of two bam-files.

5.5 Variants

vcf2vcf.py - manipulate vcf files Sort a vcf file.

5.6 Genomics

diff_chains.py - compare to chain formatted files How many residues to the same locations, do different locations, etc.

<no title> Output coverage statistics for a UCSC liftover chain file.

CHAPTER

SIX

CONTRIBUTING TO CGAT CODE

We encourage everyone who uses parts of the CGAT code collection to contribute. Contributions can take many forms: bugreports, bugfixes, new scripts and pipelines, documentation, tests, etc. All contributions are welcome.

6.1 Checklist for new scripts/modules

Before adding a new scripts to the repository, please check if the following are true:

1. The script performs a non-trivial task. If a one-line command line entry using standard unix commands can give the same effect, avoid adding a script to the repository.
2. The script has a clear purpose. Scripts should follow the unix philosophy. They should concentrate on one task and do it well. Ideally, the major input and output can be read from and written to standard input and standard output, respectively.
3. The script follows the naming convention of cgat.tools.
4. The scripts follows the *Style Guide*.
5. The script implements the `-h/--help` options. Ideally, the script has been derived from `scripts/cgat_script_template.py`.
6. The script can be imported. Ideally, it imports without performing any actions or writing output.
7. The script is well documented and the documentation has been added to the CGAT documentation. There should be an entry in `doc/scripts.rst` and a file `doc/scripts/newscript.py`.
8. The script has at least one test case added to `tests` - and the test works (see *Testing*).

6.2 Building extensions

Using pyximport, it is (relatively) straight-forward to add optimized C-code to python scripts and, for example, access pysam internals and the underlying samtools library. See for example *Purpose*.

To add an extension, the following needs to be in place:

1. The main script (`scripts/bam2stats.py`). The important lines in this script are:

```
try:
    import pyximport
    pyximport.install()
    import _bam2stats
except ImportError:
    import CGAT._bam2stats as _bam2stats
```

The snippet first attempts to build and import the extension by setting up pyximport and then importing the cython module as `_bam2stats`. In case this fails, as is the case for an installed code, it looks for a pre-built extension (by `setup.py`) in the CGAT pacakge.

2. The cython implementation `_bam2stats.pyx`. This script imports the pysam API via:

```
from csamtools cimport *
```

This statement imports, amongst others, `AlignedRead` into the namespace. Speed can be gained from declaring variables. For example, to efficiently iterate over a file, an `AlignedRead` object is declared:

```
# loop over samfile
cdef AlignedRead read
for read in samfile:
    ...
```

3. A `pyxbld` providing pyximport with build information. Required are the locations of the samtools and pysam header libraries of a source installation of pysam plus the `csamtools.so` shared library. For example:

```
def make_ext(modname, pyxfilename):
    from distutils.extension import Extension
    import pysam, os
    dirname = os.path.dirname( pysam.__file__ )[:-len("pysam")]
    return Extension(name = modname,
                     sources=[pyxfilename],
                     extra_link_args=[ os.path.join( dirname,
                            "csamtools.so")],
                     include_dirs =  pysam.get_include(),
                     define_macros = pysam.get_defines() )
```

If the script `bam2stats.py` is called the first time, pyximport will compile the cython extension `_bam2stats.pyx` and make it available to the script. Compilation requires a working compiler and cython installation. Each time `_bam2stats.pyx` is modified, a new compilation will take place.

pyximport comes with cython.

6.3 Writing recipes

Recipes are short use cases demonstrating the use of one or more CGAT utilities to address a specific problem.

Recipes should be written as ipython notebooks. The recipe notebooks are stored in the `recipes` directory in the repository. Each recipe is within its individual directory. This minimizes interference between each document, but also means that currently each notebook needs a separate notebook server to be developped.

To build all recipes, type:

```
cd recipes
make html
make clean
```

This will build html files that are deposited in the docs directory.

The last cleaning up step is important in order to remove large files created during the notebook execution.

Note: The commands above require the runipy python module. To install, type:

```
pip install runipy
```

Data for recipes can be made available in www.cgat.org/downloads/public/cgat/recipes. Ideally, recipes should make use of publicly available data sets such as ENCODE.

Attempt to add a plot to the end of a recipe, using R commands to create the plot within the notebook.

6.4 Writing pipelines

Best practice for CGAT pipelines:

1. All non-trivial code should be extracted to modules or scripts.
2. Modules should not access PARAMS dictionary directly, but parameters should be passed to the function.
3. Important processing steps where different external tools could potentially be employed the design of the module classes should be carefully considered to ensure consistent input and output file formats for different tools. PipelineMapping provides a good example for this.
4. All production pipelines should include tests for consistency which can be run automatically.
5. Where appropriate pipelines should include a small test dataset with published results for comparison. This dataset can be run on each pipeline run and included in the pipeline report where it can be used as a pipeline control.
6. Periodic code review meetings where interested parties can agree of major changes to production pipelines and associated modules – to be arranged as required.
7. The best way to manage pipeline improvements is by individuals using pipelines taking responsibility for incremental improvement. As best practice fellows should announce plans to modify particular pipelines and modules on the CGAT members list to avoid duplication of effort. Fellows should log the changes that they make in a change log and document both modules and pipelines in detail.
8. Add a section with Requirements to all pipeline scripts and tools. Only add them in files where the actual dependency arises, see *<no title>*.

CHAPTER

SEVEN

REFERENCE

This section describes the layout of the code repository and contains the API reference to the complete contents of the code collection.

7.1 Repository layout

The repository contains the following directories:

scripts Scripts in the code collection.

CGAT Modules for code shared between scripts in the collection.

doc The sphinx documentation of the code repository.

tests Testing code

recipes ipython notebook recipes and tutorials.

galaxy scripts to hook the code collection into galaxy.

7.2 API

7.2.1 Scripts

This document contains all the scripts for/by CGAT. Scripts are written to be called from the command line.

Genomics

bam2geneprofile.py - build meta-gene profile for a set of transcripts/genes

Tags Genomics NGS Genesets Intervals GTF BAM Summary

Purpose

This script takes a *gtf* formatted file, a short reads *bam* formatted file and computes meta-gene profiles over various annotations derived from the *gtf* file.

A meta-gene profile is an abstract genomic entity over which reads stored in a *bam* formatted file have been counted. A meta-gene might be an idealized eukaryotic gene (upstream, exonic sequence, downstream) or any other genomic landmark of interest such as transcription start sites.

The script can be used to visualize binding profiles of a chromatin mark in gene bodies, binding of transcription factors in promotors or sequencing bias (e.g. 3' bias) in RNA-Seq data.

This script is designed with a slight emphasis on RNA-Seq datasets. For example, it takes care of spliced reads, by using the CIGAR string in the BAM file to accurately define aligned bases (if the –base-accurate is specified, currently this feature is turned off by default for speed reasons).

Alternatively, for the purpose of visualizing binding profiles of transcription factor ChIP-Seq without the need to use any genomic annotations (ENSEMBL, or refseq), you may also consider using *bam2peakshape.py - compute peak shape features from a bam-file*, which is designed with a slight emphasis on Chip-Seq datasets. For example, *bam2peakshape.py - compute peak shape features from a bam-file* is able to center the counting window to the summit of every individual peak. *bam2peakshape.py - compute peak shape features from a bam-file* is also able to: (1) plot the control ChIP-Seq library to enable side-by-side comparison; (2) randomize the given regions to provide a semi-control.

Usage

Quick start examples

The following command will generate the gene profile plot similar to Fig 1(a) in the published cgat paper, but using a test dataset that is much smaller and simpler than the dataset used for publishing the cgat paper.

```
python ./scripts/bam2geneprofile.py
    --bam-file=./tests/bam2geneprofile.py/
↪multipleReadsSplicedOutAllIntronsAndSecondExon.bam
    --gtf-file=./tests/bam2geneprofile.py/onegeneWithoutAnyCDS.gtf.gz
    --method=geneprofile
    --reporter=gene
```

In the following, a slightly more involved example will use more features of this script. The following command generate the gene profile showing base accuracy of upstream (500bp), exons, introns and downstream(500bp) of a gene model from some user supplied RNA-Seq data and geneset.

```
python ./scripts/bam2geneprofile.py
    --bam-file=./rnaseq.bam
    --gtf-file=./geneset.gtf.gz
    --method=geneprofilewithintrons
    --reporter=gene
    --extension-upstream=500
    --resolution-upstream=500
    --extension-downstream=500
    --resolution-downstream=500
```

The output will contain read coverage over genes. The profile will contain four separate segments:

1. the upstream region of a gene (set to be 500bp), (`--extension-upstream=500`).
2. the transcribed region of a gene. The transcribed region of every gene will be scaled to 1000 bp (default), shrinking longer transcripts and expanding shorter transcripts.

3. the intronic regions of a gene. These will be scaled to 1000b (default).
4. the downstream region of a gene (set to be 500bp), (`--extension-downstream=500`).

Detailed explaination

The `bam2geneprofile.py` script reads in a set of transcripts from a *gtf* formatted file. For each transcript, overlapping reads from the provided *bam* file are collected. The counts within the transcript are then mapped onto the meta-gene structure and counts are aggregated over all transcripts in the *gtf* file.

Bam files need to be sorted by coordinate and indexed.

A meta-gene structure has two components - regions of variable size, such as exons, introns, etc, which nevertheless have a fixed start and end coordinate in a transcript. The other component are regions of fixed width, such a regions of a certain size upstream or downstream of a landmark such as a transcription start site.

The size of the former class, regions of variable size, can be varied with `--resolution` options. For example, the option `--resolution-upstream-utr=1000` will create a meta-gene with a 1000bp upstream UTR region. UTRs that are larger will be compressed, and UTRs that are smaller, will be stretched to fit the 1000bp meta-gene UTR region.

The size of fixed-width regions can be set with `--extension` options. For example, the options `--extension-upstream` will set the size of the uptsream extension region to 1000bp. Note that no scaling is required when counting reads towards the fixed-width meta-gene profile.

Type:

```
python bam2geneprofile.py --help
```

for command line help.

Options

The script provides a variety of different meta-gene structures i.e. geneprofiles, selectable via using the option: (`--method`).

Profiles

Different profiles are accessible through the `--method` option. Multiple methods can be applied at the same time. While `upstream` and `downstream` typically have a fixed size, the other regions such as `CDS`, `UTR` will be scaled to a common size.

utrprofile UPSTREAM - UTR5 - CDS - UTR3 - DOWNSTREAM gene models with UTR. Separate the coding section from the non-coding part.

geneprofile UPSTREAM - EXON - DOWNSTREAM simple exonic gene models

geneprofilewithintrons UPSTREAM - EXON - INTRON - DOWNSTREAM

gene models containing also intronic sequence, only correct if used with `--use-base-accuracy` option.

separateexonprofile UPSTREAM - FIRST EXON - EXON - LAST EXON - DOWNSTREAM

gene models with the first and last exons separated out from all other exons. Only applicable to gene models with > 1 exons.

separateexonprofilewithintrons UPSTREAM - FIRST EXON - EXON - INTRON - LAST EXON - DOWNSTREAM

gene models with first and last exons separated out, and includes all introns together. Excludes genes with < 2 exons and no introns.

geneprofileabsolutedistancefromthreeprimeend

UPSTREAM - EXON (absolute distance, see below) - INTRON (absolute distance, see below) - DOWNSTREAM (the downstream of the exons) region, the script counts over the mRNA transcript only, skipping introns. Designed to visualize the 3 prime bias in RNASeq data, only correct if used together with `--use-base-accuracy` option.

absolute distance: In order to to visualize the 3 prime bias, genes are not supposed to be streched to equal length as it did in all other counting methods. In this counting method, we first set a fix length using `--extension-exons-absolute-distance-topolya`, the script will discard genes shorter than this fixed length. For genes (when all the exons stitched together) longer than this fixed length, the script will only count over this fixed length (a absolute distance) from three prime end, instead of compress the longer genes. Same goes for absolute distance intron counting.

tssprofile UPSTREAM - DOWNSTREAM transcription start/stop sites

intervalprofile UPSTREAM - INTERVAL - DOWNSTREAM Similar to geneprofile, but count over the complete span of the gene (including introns).

midpointprofile UPSTREAM - DOWNSTREAM aggregate over midpoint of gene model

Normalization

Normalization can be applied in two stages of the computation.

Count vector normalization

Before adding counts to the meta-gene profile, the profile for the individual transcript can be normalized. Without normalization, highly expressed genes will contribute more to the meta-gene profile than lowly expressed genes. Normalization can assure that each gene contributes an equal amount.

Normalization is applied to the vector of read counts that is computed for each transcript. Normalization can be applied for the whole transcript (`total`) or on a per segment basis depending on the counter. For example, in the gene counter, exons, upstream and downstream segments can be normalized independently.

Counts can be normalized either by the maximum or the sum of all counts in a segment or across the whole transcript. Normalization is controlled with the command line option `--normalize-trancript`. Its arguments are:

- `none`: no normalization
- `sum`: sum of counts within a region (exons, upstream, ...). The area under the curve will sum to 1 for each region.
- `max`: maximum count within a region (exons,upstream, ...).
- `total-sum`: sum of counts across all regions. The area under the curve will sum to 1 for the complete transcript.
- `total-max`: maximum count across all regions.

The options above control the contribution of individual transcripts to a meta-gene profile and are thus suited for example for RNA-Seq data.

The options above do not control for different read-depths or any local biases. To compare meta-gene profiles between samples, additional normalization is required.

Meta-gene profile normalization

To enable comparison between experiments, the meta-gene profile itself can be normalized. Normalization a profile can help comparing the shapes of profiles between different experiments independent of the number of reads or transcripts used in the construction of the meta-gene profile.

Meta-gene profile normalization is controlled via the `--normalize-profile` option. Possible normalization are:

- none: no normalization
- area: normalize such that the area under the meta-gene profile is 1.
- counts: normalize by number of features (genes,tss) that have been counted.
- background: normalize with background (see below).

A special normalization is activated with the `background` option. Here, the counts at the left and right most regions are used to estimate a background level for each transcript. The counts are then divided by this background-level. The assumption is that the meta-gene model is computed over a large enough area to include genomic background.

Genes versus transcripts

The default is to collect reads on a per-transcript level. Alternatively, the script can merge all transcripts of a gene into a single virtual transcript. Note that this virtual transcript might not be a biologically plausible transcript. It is usually better to provide `bam2geneprofile.py` with a set of representative transcripts per gene in order to avoid up-weighting genes with multiple transcripts.

Control

If control files (chip-seq input tracks) are supplied, counts in the control file can be used to compute a fold-change.

Bed and wiggle files

The densities can be computed from *bed* or *wiggle* formatted files. If a *bed* formatted file is supplied, it must be compressed with and indexed with `tabix`.

Note: Paired-endedness is ignored. Both ends of a paired-ended read are treated individually.

Command line options

```
usage: bam2geneprofile [-h] [--version]
                       [-m {geneprofile,tssprofile,utrprofile,intervalprofile,
↪midpointprofile,geneprofilewithintrons,geneprofileabsolutedistancefromthreeprimeend,
↪separateexonprofile,separateexonprofilewithintrons}]
                       [-b BAM] [-c BAM] [-g GTF]
                       [--normalize-transcript {none,max,sum,total-max,total-sum}]
                       [--normalize-profile {all,none,area,counts,background}]
```

(continues on next page)

(continued from previous page)

```
                        [-r {gene,transcript}] [-i SHIFTS] [-a] [-u]
                        [-e EXTENDS]
                        [--resolution-upstream RESOLUTION_UPSTREAM]
                        [--resolution-downstream RESOLUTION_DOWNSTREAM]
                        [--resolution-upstream-utr RESOLUTION_UPSTREAM_UTR]
                        [--resolution-downstream-utr RESOLUTION_DOWNSTREAM_UTR]
                        [--resolution-cds RESOLUTION_CDS]
                        [--resolution-first-exon RESOLUTION_FIRST]
                        [--resolution-last-exon RESOLUTION_LAST]
                        [--resolution-introns RESOLUTION_INTRONS]
                        [--resolution-exons-absolute-distance-topolya RESOLUTION_EXONS_
↪ABSOLUTE_DISTANCE_TOPOLYA]
                        [--resolution-introns-absolute-distance-topolya RESOLUTION_
↪INTRONS_ABSOLUTE_DISTANCE_TOPOLYA]
                        [--extension-exons-absolute-distance-topolya EXTENSION_EXONS_
↪ABSOLUTE_DISTANCE_TOPOLYA]
                        [--extension-introns-absolute-distance-topolya EXTENSION_
↪INTRONS_ABSOLUTE_DISTANCE_TOPOLYA]
                        [--extension-upstream EXTENSION_UPSTREAM]
                        [--extension-downstream EXTENSION_DOWNSTREAM]
                        [--extension-inward EXTENSION_INWARD]
                        [--extension-outward EXTENSION_OUTWARD]
                        [--scale-flank-length SCALE_FLANKS]
                        [--control-factor CONTROL_FACTOR]
                        [--output-all-profiles]
                        [--counts-tsv-file INPUT_FILENAME_COUNTS]
                        [--background-region-bins BACKGROUND_REGION_BINS]
                        [--output-res RESOLUTION_IMAGES]
                        [--image-format IMAGE_FORMAT] [--timeit TIMEIT_FILE]
                        [--timeit-name TIMEIT_NAME] [--timeit-header]
                        [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                        [--log-config-filename LOG_CONFIG_FILENAME]
                        [--tracing {function}] [-? ?]
                        [-P OUTPUT_FILENAME_PATTERN] [-F] [-I STDIN]
                        [-L STDLOG] [-E STDERR] [-S STDOUT]
bam2geneprofile: error: argument -?: expected one argument
```

bed2bed - manipulate bed files

Purpose

This script provides various methods for merging (by position, by name or by score), filtering and moving bed formatted intervals and outputting the results as a bed file

This script provides several methods, each with a set of options

to control behavoir:

cgat.tools.bed2bed.**merge**()

+++++

Merge together overlapping or adjacent intervals. The basic

functionality is similar to bedtools merge, but with some additions:

*** Merging by name: specifying the --merge-by-name option will mean**
that only overlaping (or adjacent intervals) with the same value in the 4th column of the bed will be merged

*** Removing overlapping intervals with inconsistent names: set the**
`--remove-inconsistent-names` option.

.. caution::
Intervals of the same name will only be merged if they are consecutive in the bed file.

*** Only output merged intervals: By specifiying the --merge-min-intervals=n**
options, only those intervals that were created by merging at least n intervals together will be output

Intervals that are close but not overlapping can be merged by setting

--merge-distance to a non-zero value

`cgat.tools.bed2bed.`**`bins`**`()`

++++

Merges together overlapping or adjecent intervals only if they have

"similar" scores. Score similarity is assessed by creating a number of

score bins and assigning each interval to a bin. If two adjacent

intervals are in the same bin, the intervals are merged. Note that in

contrast to merge-by-name above, two intervals do not need to be

overlapping or within a certain distance to be merged.

There are several methods to create the bins:

*** equal-bases: Bins are created to that they contain the same number of bases.**
Specified by passing "equal-bases" to –binning-method. This is the default.

*** equal-intervals: Score bins are create so that each bin contains the**
same number of intervals. Specified by passing "equal-intervals" to –binning-method.

*** equal-range: Score bins are created so that**
each bin covers the same fraction of the total range of scores. Specified by passing "equal-range" to –binning-method.

*** bin-edges: Score binds can be specified by manually passing a comma**
seperated list of bin edges to –bin-edges.

The number of bins is specified by the --num-bins options, and the

default is 5.

`cgat.tools.bed2bed.`**`block`**`()`

+++++

Creates blocked bed12 outputs from a bed6, where intervals with the

same name are merged together to create a single bed12 entry.

.. Caution:: Input must be sorted so that entries of the same

name are together.

filter-genome

++++++++++++

Removes intervals that are on unknown contigs or extend off the 3' or

5' end of the contig. Requires a tab seperated input file to -g which lists the contigs in the genome, plus their lengths.

sanitize-genome

++++++++++++++

As above, but instead of removing intervals overlapping the ends of contigs, truncates them. Also removes empty intervals.

filter-names

++++++++++++

Output intervals whose names are in list of desired names. Names are supplied as a file with one name on each line.

cgat.tools.bed2bed.**shift**()

+++++

Moves intervals by the specified amount, but will not allow them to be shifted off the end of contigs. Thus if a shift will shift the start of end of the contig, the interval is only moved as much as is possible without doing this.

rename-chr

++++++++++

Renames chromosome names. Source and target names are supplied as a file with two columns. Examples are available at:

https://github.com/dpryan79/ChromosomeMappings

Note that unmapped chromosomes are dropped from the output file.

Other options

++++++++++++

-g/--genome-file, -b/--bam-file:
the filter-genome, sanitize-genome and shift methods require a genome in order to ensure they are not placing intervals outside the limits of contigs. This genome can be supplied either as a samtools or cgat indexed genome, or extracted from the header of a bam file.

Examples

Merge overlapping or adjectent peaks from a CHiP-seq experiment where the intervals have the same name:

cat chip-peaks.bed | cgat bed2bed –method=merge –merge-by-name > chip-peaks-merged.bed

Merge adjected ChIP-seq peaks if their scores are in the same quartile of all scores:

cat chip-peaks.bed | cgat bed2bed –method=bins –binning-method=equal-intervals –num-bins=4

Remove intervals that overlap the ends of a contig and those that are on a non-standard contig. Take the input intervals from a file rather than stdin. Note that hg19.fasta has been indexed with *index_genome*:

cgat bed2bed –method=filter-genome –genome-file=hg19.fasta -I chip-peaks.bed -O chip-peaks-sanitized.bed

Convert a bed file contain gene structures with one line per exon to a bed12 with linked block representing the gene structure. Note the transparent use of compressed input and output files:

cgat bed2bed –method=block -I transcripts.bed.gz -O transcripts.blocked.bed.gz

Rename UCSC chromosomes to ENSEMBL.

cat ucsc.bed | cgat bed2bed –method=rename-chr –rename-chr-file=ucsc2ensembl.txt > ensembl.bed

Usage

cgat bed2bed –method=[METHOD] [OPTIONS]

Will read bed file from stdin and apply the specified method

Command line options

```
usage: bed2bed [-h]
               [-m {merge,filter-genome,bins,block,sanitize-genome,shift,extend,
↪filter-names,rename-chr}]
               [--num-bins NUM_BINS] [--bin-edges BIN_EDGES]
               [--binning-method {equal-bases,equal-intervals,equal-range}]
               [--merge-distance MERGE_DISTANCE]
               [--merge-min-intervals MERGE_MIN_INTERVALS] [--merge-by-name]
               [--merge-and-resolve-blocks] [--merge-stranded]
               [--remove-inconsistent-names] [--offset OFFSET]
               [-g GENOME_FILE] [-b BAM_FILE] [--filter-names-file NAMES]
               [--rename-chr-file RENAME_CHR_FILE] [--timeit TIMEIT_FILE]
               [--timeit-name TIMEIT_NAME] [--timeit-header]
               [--random-seed RANDOM_SEED] [-v LOGLEVEL]
               [--log-config-filename LOG_CONFIG_FILENAME]
               [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
               [-E STDERR] [-S STDOUT]
bed2bed: error: argument -?: expected one argument
```

bed2gff.py - convert bed to gff/gtf

Tags Genomics Intervals BED GFF Conversion

Purpose

This script converts a *bed*-formatted file to a *gff* or *gtf*-formatted file.

It aims to populate the appropriate fields in the *gff* file with columns in the *bed* file.

If `--as-gtf` is set and a name column in the *bed* file is present, its contents will be set as `gene_id` and `transcript_id`. Otherwise, a numeric `gene_id` or `transcript_id` will be set according to `--id-format`.

Usage

Example:

```
# Preview input bed file
zcat tests/bed2gff.py/bed3/bed.gz | head
# Convert BED to GFF format
cgat bed2gff.py < tests/bed2gff.py/bed3/bed.gz > test1.gff
# View converted file (excluding logging information)
cat test1.gtf | grep -v "#" | head
```

chr1	bed	exon	501	1000	.	.	.	gene_id “None”; transcript_id “None”;
chr1	bed	exon	15001	16000	.	.	.	gene_id “None”; transcript_id “None”;

Example:

```
# Convert BED to GTF format
cgat bed2gff.py --as-gtf < tests/bed2gff.py/bed3/bed.gz > test2.gtf
# View converted file (excluding logging information)
cat test2.gtf | grep -v "#" | head
```

chr1	bed	exon	501	1000	.	.	.	gene_id “00000001”; transcript_id “00000001”;
chr1	bed	exon	15001	16000	.	.	.	gene_id “00000002”; transcript_id “00000002”;

Type:

```
cgat bed2gff.py --help
```

for command line help.

Command line options

```
usage: bed2gff [-h] [-a] [-f ID_FORMAT] [--timeit TIMEIT_FILE]
               [--timeit-name TIMEIT_NAME] [--timeit-header]
               [--random-seed RANDOM_SEED] [-v LOGLEVEL]
               [--log-config-filename LOG_CONFIG_FILENAME]
               [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
               [-E STDERR] [-S STDOUT]
bed2gff: error: argument -?: expected one argument
```

gff2bed.py - convert from gff/gtf to bed

Tags Genomics Intervals GFF BED Conversion

Purpose

This script converts GFF or GTF formatted files to BED formatted files.

Documentation

Users can select the field from the GTF file to be used in the name field of the BED file using `--set-name`. Choices include "gene_id", "transcript_id", "class", "family", "feature", "source", "repName" and "gene_biotype". To specify the input is in GTF format use –is-gtf.

BED files can contain multiple tracks. If required, users can use the "feature" or "source" fields in the input GFF file to specifiy different tracks in the BED file (default none).

Usage

Example:

```
# View input GTF file
head tests/gff2bed.py/mm9_ens67_geneset_100.gtf

# Convert GTF to bed format using gene_id as name and group by GTF feature
cat tests/gff2bed.py/mm9_ens67_geneset_100.gtf | cgat gff2bed.py --is-gtf --set-
↪name=gene_id --track=feature > mm9_ens67_geneset_100_feature.bed
```

track name=CDS					
chr18	3122494	3123412	ENSMUSG0000009	91539	•
chr18	3327491	3327535	ENSMUSG0000006	63889	•
chr18	3325358	3325476	ENSMUSG0000006	63889	•

Command line options

```
usage: gff2bed [-h] [--is-gtf]
               [--set-name {gene_id,transcript_id,class,family,feature,source,repName,
↪gene_biotype}]
               [--track {feature,source,None}] [--bed12-from-transcripts]
               [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
               [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
               [--log-config-filename LOG_CONFIG_FILENAME]
               [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
               [-E STDERR] [-S STDOUT]
gff2bed: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
```

(continues on next page)

(continued from previous page)

```
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'psl2assembly'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'psl2psl'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'psl2gff'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'psl2map'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'psl2wiggle'
```

diff_gtf.py - compute overlap between multiple gtf files

Tags Genomics Intervals Genesets GTF Comparison

Purpose

This script compares multiple set of gtf files. It computes the overlap between bases, exons and genes between each pair of gtf files.

If results from a previous run are present, existing pairs are not re-computed but simply echoed.

The output is a tab-separated table with counts for each pair of files being compared. The fields are:

Column	*Content*
set	Name of the set
ngenes_total	number of genes in set
ngenes_ovl	number of genes overlapping
ngenes_unique	number of unique genes
nexons_total	number of exons in set
nexons_ovl	number of exons overlapping
nexons_unique	number of unique exons
nbases_total	number of bases in gene set
nbases_ovl	number of bases overlapping
nbases_unique	number of unique bases

Each of these fields will appear twice, once for each of the pair of files. Hence ngenes_unique1 will be the number of genes in set1 that have no exons that overlap with any exons in set2, and vice versa for ngenes_unique2. And on for each field in the table above. This makes a total of 9*2=18 fields containing counts, each starting with an n.

A further set of 18 fields each start with a `p` and are the corresponding percentage values.

Options

-s, --ignore-strand Ignore strand infomation so that bases overlap even if exons/genes are on different strands

-u, --update=FILENAME Read in previous results from FILENAME and only output comparisons that are missing.

-p, --pattern-identifier=PATTERN Provide a regular expression pattern for converting a filename into a set name for the output. The regular expression should capture at least one group. That group will be used to identify that file in the output table (see examples)

Examples

For example if we have two gtf_files that look like:

```
first_set_of_genes.gtf:
1    protein_coding  exon    1       10      .       +       .       gene_id "1";␣
↪transcript_id "1"
1    protein_coding  exon    20      30      .       +       .       gene_id "1";␣
↪transcript_id "1"

second_set_of_genes.gtf:
1    protein_coding  exon    25      35      .       +       .       gene_id "1";␣
↪transcript_id "1"
2    protein_coding  exon    100     200     .       +       .       gene_id "2";␣
↪transcript_id "3"
```

Then the command:

```
python diff_gtf.py *.gtf --pattern-identifier='(.+)_of_genes.gtf' > out.tsv
```

would produce an output file that has a single row with set1 being "second_set" and set2 being "first_set" (these are extracted using that –pattern-identifier option). It will report that set1 contains 2 genes and set2 1 gene. That for each set one of these genes overlaps with the other set. For set1 it will report that 1 gene is unique and that no genes are unique for set2 and so on for exons and bases.

If we want to add a third file to the comparison, "third_set_of_genes.gtf", we don't need to redo the comparison between first_set_of_genes.gtf and second_set_of_genes.gtf:

```
python diff_gtf.py --update=out.tsv *.gtf.gz > new.tsv
```

This will output a table with a row for third_set vs second_set and third_set vs second_set, along with the comparison of first_set and second_set that will simply be copied from the previous results. It is important to include all files on the command line that are to be output. Any comparisons in `out.tsv` that correspond to files that are not given on the command line will not be output.

Usage

:: cgat diff_gtf.py GTF GTF [GTF [GTF [...]]] [OPTIONS] cgat diff_gtf GTF1 –update=OUTFILE [OPTIONS]

where GTF is a gtf or compressed gtf formated file and OUTFILE is the results from a previous run. At least two must be provided unless –update is present.

Type:

```
python diff_gtf.py --help
```

for command line help.

Command line options

```
usage: diff-gtf [-h] [--version] [-s] [-u FILENAME_UPDATE] [-p PATTERN_ID]
                [-g] [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                [--log-config-filename LOG_CONFIG_FILENAME]
                [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                [-E STDERR] [-S STDOUT]
diff-gtf: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'fasta2gff'
```

gff2psl.py - convert from gff to psl

Tags Genomics Intervals GFF PSL Conversion

Purpose

This scripts converts from a *gff* formatted file to a *psl* formatted file. The output can be modified by the following command line options:

--allow-duplicates	keep duplicate entries from gff/gtf input file
--genome-file	restrict output to gff/gtf entries with contigs in fasta file
--queries-tsv-file	restrict output to queries in fasta file

Usage

Example:

```
python gff2psl.py < in.gff > out.psl
```

Type:

```
python gff2psl.py --help
```

for command line help. genome-file

Command line options

```
usage: gff2psl [-h] [--is-gtf] [--no-header] [-g GENOME_FILE]
               [--queries-tsv-file INPUT_FILENAME_QUERIES]
               [--allow-duplicates] [--timeit TIMEIT_FILE]
               [--timeit-name TIMEIT_NAME] [--timeit-header]
               [--random-seed RANDOM_SEED] [-v LOGLEVEL]
               [--log-config-filename LOG_CONFIG_FILENAME]
               [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
               [-E STDERR] [-S STDOUT]
gff2psl: error: argument -?: expected one argument
```

gff2coverage.py - compute genomic coverage of gff intervals

Tags Genomics Intervals Summary GFF

Purpose

This script computes the genomic coverage of intervals in a *gff* formatted file. The coverage is computed per feature.

Usage

You can use two methods to compute the coverage: **genomic** and **histogram**.

Let us explain their usage with this `small.gtf` file:

```
19 processed_transcript exon 16 16 . - . gene_id
19 processed_transcript exon 27 27 . - . gene_id
19 processed_transcript exon 8  8  . - . gene_id
19 processed_transcript exon 19 19 . - . gene_id
19 processed_transcript exon 5  5  . - . gene_id
```

and this toy example (`small.fasta`) of an indexed fasta file:

```
>chr19
GCCGGCCTCTACCTGCAGCAGATGCCCTAT
```

Both files (`small.gtf` and `small.fasta`) are included in the GitHub repository.

genomic method

The **genomic** method computes the coverage of intervals accross the genome file given as input. Let us see how to apply the genomic method to the small examples above:

```
python gff2coverage.py --method=genomic --genome-file=small < small.gtf
```

The output (wrapped to fit here) will be:

```
contig  source  feature  intervals  bases  p_coverage  total_p_coverage
19      trans.  exon     5          5      16.666667   16.666667
```

As you can see the information displayed is the following: contig name, source, feature name, number of intervals within the contig, number of bases, percentage of coverage in the contig, and percentage of coverage in the genome file.

histogram method

On the contrary, if you want to compute the coverage of intervals within the *gff* file itself summarized as an histogram and grouped by contig name, please use the histogram method.

To use the histogram method with the input files above, please type:

```
python gff2coverage.py --method=histogram --window=5 --features=exon --output-
↪filename-pattern=%s.hist < small.gtf
```

In this case the output (written to file `19.hist`) is:

```
abs_pos  rel_pos  abs_exon  rel_exon
0        0.0000   1         0.2000
5        0.1852   2         0.4000
10       0.3704   2         0.4000
15       0.5556   4         0.8000
20       0.7407   4         0.8000
25       0.9259   5         1.0000
```

The output is given as a pair of columns. The first pair of columns always appears and lists the cumulative numbers of nucleotides in each window or bin –absolute and relative values in the former and latter columns, respectively. The subsequent pair of columns depends on the values given to the `--features` option. In this example there is an extra column for the `exon` feature but you could especify as many of them as you wanted among those features listed in your *gff* file.

On the other hand, the `--num-bins` option can be used instead of `--window` along with `--genome-file` to define the number of bins for the resultant histogram. This parameter is used by default (with value: 1000) when using the histogram method.

Please note the following:

- you need to specify the feature name explicitly (with the `--feature` option) to compute the genomic coverage of that feature. You can also usea comma-separated list of feature names.
- the output of the histogram method goes to a file (in the current workingdirectory) which is named as the contig name by default. To change thisbehaviour, please use the `--output-filename-pattern` option where `%s` will be substituted by the contig name.

Command line options

```
usage: gff2coverage [-h] [--version] [-g GENOME_FILE] [-f FEATURES]
                    [-w WINDOW_SIZE] [-b NUM_BINS] [-m {genomic,histogram}]
                    [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                    [--timeit-header] [--random-seed RANDOM_SEED]
                    [-v LOGLEVEL] [--log-config-filename LOG_CONFIG_FILENAME]
                    [--tracing {function}] [-? ?] [-P OUTPUT_FILENAME_PATTERN]
                    [-F] [-I STDIN] [-L STDLOG] [-E STDERR] [-S STDOUT]
gff2coverage: error: argument -?: expected one argument
```

gff2fasta.py - output sequences from genomic features

Tags Genomics Intervals Sequences GFF Fasta Transformation

Purpose

This script outputs the genomic sequences for intervals within a *gff* or :term: *gtf* formatted file.

The ouput can be optionally masked and filtered.

Usage

If you want to convert a `features.gff` file with intervals information into a *fasta* file containing the sequence of each interval, use this script as follows:

```
python gff2fasta.py --genome-file=hg19 < features.gff > features.fasta
```

The input can also be a *gtf* formatted file. In that case, use the `--is-gtf` option:

```
python gff2fasta.py --genome-file=hg19 --is-gtf < features.gtf > features.fasta
```

If you want to add a polyA tail onto each transcript you can use the *extend* options:

> python gff2fasta.py –genome-file=hg19 –is-gtf –extend-at=3 –extend-by=125 –extend-with=A < features.gtf > features.fasta

If you want to merge the sequence of similar features together, please use `--merge-overlapping`:

```
python gff2fasta.py --genome-file=hg19 --merge-overlapping < features.gff > features.
↪fasta
```

It is possible to filter the output by selecting a minimum or maximum number of nucleotides in the resultant fasta sequence with `--max-length` or `--min-interval-length` respectively:

```
python gff2fasta.py --genome-file=hg19 --max-length=100 < features.gff > features.
↪fasta
```

Or you can also filter the output by features name with the `--feature` option:

```
python gff2fasta.py --genome-file=hg19 --feature=exon < features.gff > features.fasta
```

On the other hand, low-complexity regions can be masked with the `--masker` option and a given *gff* formatted file:

```
python gff2fasta.py --genome-file=hg19 --masker=dust --maskregions-bed-file=intervals.
↪gff < features.gff > features.fasta
```

where `--masker` can take the following values: `dust`, `dustmasker`, and `softmask`.

Options

--is-gtf Tells the script to expect a *gtf* format file

--genome-file PATH to Fasta file of genome build to use

--merge-overlapping Merge features in *gtf*/*gff* file that are adjacent and share attributes

--method=filter --filter-method Filter on a *gff* feature such as `exon` or `CDS`

--maskregions-bed-file Mask sequences in intervals in *gff* file

--remove-masked-regions Remove sequences in intervals in *gff* file rather than masking them

--min-interval-length Minimum output sequence length

--max-length Maximum output sequence length

--extend-at Extend sequence at 3', 5' or both end. Optionally '3only' or '5only' will return only the 3' or 5' extended sequence

--extend-by Used in conjunction with `--extend-at`, the number of nucleotides to extend by

--extend-with Optional. Used in conjunction with `--extend-at` and `--extend-by`. Instead of extending by the genomic sequence, extend by this string repeated n times, where n is –entend-by

--masker Masker type to use: dust, dustmasker, soft or none

--fold-at Fold the fasta sequence every n bases

--naming-attribute Use this attribute to name the fasta entries

Command line options

```
usage: gff2fasta [-h] [--is-gtf] [-g GENOME_FILE] [-m] [-e FEATURE] [-f gff]
                 [--remove-masked-regions] [--min-interval-length MIN_LENGTH]
                 [--max-length MAX_LENGTH]
                 [--extend-at {none,3,5,both,3only,5only}]
                 [--header-attributes] [--extend-by EXTEND_BY]
                 [--extend-with EXTEND_WITH]
                 [--masker {dust,dustmasker,softmask,none}]
                 [--fold-at FOLD_AT] [--fasta-name-attribute NAMING_ATTRIBUTE]
                 [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                 [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                 [--log-config-filename LOG_CONFIG_FILENAME]
                 [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                 [-E STDERR] [-S STDOUT]
gff2fasta: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'gtf2alleles'
```

gff2gff.py - manipulate gff files

Tags Genomics Intervals GFF Manipulation

Purpose

This scripts reads a *gff* formatted file, applies a transformation and outputs the new intervals in *gff* format. The type of transformation chosen is given through the *–method`* option. Below is a list of available transformations:

`complement-groups`

output the complenent intervals for the features in the file, for example to output introns from exons. The option `--group-field` sets field/attribute to group by, e.g gene_id, transcript_id, feature, source.

`combine-groups`

combine all features in a group into a single interval. The option `--group-field` sets field/attribute to group by, see alse `complement-groups`.

`to-forward-coordinates`

translate all features forward coordinates.

`to-forward-strand`

convert to forward strand

`add-upstream-flank/add-downstream-flank/add-flank`

add an upstream/downstream flanking segment to first/last exon of a group. The amount added is given through the options `--extension-upstream` and `--extension-downstream`. If `--flank-method` is `extend`, the first/last exon will be extended, otherwise a new feature will be added.

`crop`

crop features according to features in a separate gff file. If a feature falls in the middle of another, two entries will be output."""")

`crop-unique`

remove non-unique features from gff file.

`merge-features`

merge consecutive features.

`join-features`

group consecutive features.

`filter-range` extract features overlapping a chromosomal range. The range can be set by the `--filter-range` option.

`sanitize` reconcile chromosome names between ENSEMBL/UCSC or with an indexed genomic fasta file (see *index_fasta.py - Index fasta formatted files*). Raises an exception if an unknown contig is found (unless `--skip-missing` is set). The method to sanitize is specified by `--sanitize-method`.The method to sanitize is specified by `--sanitize-method`. Options for `` `--sanitize-method` `` include "ucsc", "ensembl", "genome". A pattern of contigs to remove can be given in the option `--contig-pattern`. If `--sanitize-method` is set to `ucsc` or `ensembl`, the option `--assembly-report` is required to allow for accurate mapping between UCSC and Ensembl. If not found in the assembly report the contig names are forced into the desired convention, either by removing or prepending `chr`, this is useful for *gff* files with custom contigs. The Assembly Report can be found on the NCBI assembly page under the link "Download the

full sequence report". If `--sanitize-method` is set to `genome`, the genome file has to be provided via the option `--genome-file` or `--contigs-tsv-file`

`skip-missing`

skip entries on missing contigs. This prevents exception from being raised

`filename-agp`

agp file to map coordinates from contigs to scaffolds

`rename-chr`

Renames chromosome names. Source and target names are supplied as a file with two columns. Examples are available at: https://github.com/dpryan79/ChromosomeMappings Note that unmapped chromosomes are dropped from the output file.

Usage

For many downstream applications it is helpful to make sure that a *gff* formatted file contains only features on placed chromosomes.

As an example, to sanitise hg38 chromosome names and remove chromosome matching the regular expression patterns "ChrUn", "_alt" or "_random", use the following:

cat in.gff | gff2gff.py –method=sanitize –sanitize-method=ucsc

–assembly-report=/path/to/file –skip-missing

gff2gff.py –remove-contigs="chrUn,_random,_alt" > gff.out

The "–skip-missing" option prevents an exception being raised if entries are found on missing chromosomes

Another example, to rename UCSC chromosomes to ENSEMBL.

cat ucsc.gff | gff2gff.py –method=rename-chr

–rename-chr-file=ucsc2ensembl.txt > ensembl.gff

Type:

```
cgat gff2gff --help
```

for command line help.

Command line options

```
usage: gff2gff [-h] [--version]
               [-m {add-flank,add-upstream-flank,add-downstream-flank,crop,crop-
↪unique,complement-groups,combine-groups,filter-range,join-features,merge-features,
↪sanitize,to-forward-coordinates,to-forward-strand,rename-chr}]
               [--ignore-strand] [--is-gtf] [-c INPUT_FILENAME_CONTIGS]
               [--agp-file INPUT_FILENAME_AGP] [-g GENOME_FILE]
               [--crop-gff-file FILENAME_CROP_GFF] [--group-field GROUP_FIELD]
               [--filter-range FILTER_RANGE]
               [--sanitize-method {ucsc,ensembl,genome}]
```

(continues on next page)

(continued from previous page)

```
                [--flank-method {add,extend}] [--skip-missing]
                [--contig-pattern CONTIG_PATTERN]
                [--assembly-report ASSEMBLY_REPORT]
                [--assembly-report-hasids ASSEMBLY_REPORT_HASIDS]
                [--assembly-report-ucsccol ASSEMBLY_REPORT_UCSCCOL]
                [--assembly-report-ensemblcol ASSEMBLY_REPORT_ENSEMBLCOL]
                [--assembly-extras ASSEMBLY_EXTRAS]
                [--extension-upstream EXTENSION_UPSTREAM]
                [--extension-downstream EXTENSION_DOWNSTREAM]
                [--min-distance MIN_DISTANCE] [--max-distance MAX_DISTANCE]
                [--min-features MIN_FEATURES] [--max-features MAX_FEATURES]
                [--rename-chr-file RENAME_CHR_FILE] [--timeit TIMEIT_FILE]
                [--timeit-name TIMEIT_NAME] [--timeit-header]
                [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                [--log-config-filename LOG_CONFIG_FILENAME]
                [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                [-E STDERR] [-S STDOUT]
gff2gff: error: argument -?: expected one argument
```

gff2histogram.py - compute histograms from intervals in gff or bed format

Tags Genomics Intervals GFF Summary

Purpose

This script computes distributions of interval sizes, intersegmental distances and interval overlap from a list of intervals in *gff* or *bed* format.

The output will be written into separate files. Filenames are given by `--ouput-filename-pattern`.

Available methods are:

hist Output a histogram of interval sizes and distances between intervals in nucleotides.

stats Output summary statistics of interval sizes and distances between intervals

values Output distances, sizes, and overlap values to separate files.

all all of the above.

Usage

For example, a small gff file such as this (note that intervals need to be sorted by position):

```
chr19   processed_transcript    exon    60105   60162   .       -       .
chr19   processed_transcript    exon    60521   60747   .       -       .
chr19   processed_transcript    exon    65822   66133   .       -       .
chr19   processed_transcript    exon    66346   66416   .       -       .
chr19   processed_transcript    exon    66346   66509   .       -       .
```

will give when called as:

```
cgat gff2histogram < in.gff
```

the following output files:

hist Histogram of feature sizes and distances between adjacent features

residues	size	distance
58.0	1	na
71.0	1	na
164.0	1	na
212.0	na	1
227.0	1	na
312.0	1	na
358.0	na	1
5074.0	na	1

stats

Summary statistics of the distribution of feature size and distance between adjacent features.

data	nval	min	max	mean	median	stddev	sum	q1	q3
size	5	58.0000	312.0000	166.4000	164.0000	95.6339	832.0000	71.0000	227.0000
distance	3	212.0000	5074.0000	1881.3333	358.0000	2258.3430	5644.0000	212.0000	5074.0000

overlaps

A file with features that overlap other features, here:

```
chr19   processed_transcript    exon    66346   66416   .       -       .
↪   chr19   processed_transcript    exon    66346   66509   .       -
↪.
```

Type:

```
python gff2histogram.py --help
```

for command line help.

Command line options

```
usage: gff2histogram [-h] [--version] [-b BIN_SIZE] [--min-value MIN_VALUE]
                     [--max-value MAX_VALUE] [--no-empty-bins]
                     [--with-empty-bins] [--ignore-out-of-range]
                     [--missing-value MISSING_VALUE] [--use-dynamic-bins]
                     [--format {gff,gtf,bed}]
                     [--method {all,hist,stats,overlaps,values}]
                     [--output-section {all,size,distance}]
                     [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                     [--timeit-header] [--random-seed RANDOM_SEED]
                     [-v LOGLEVEL] [--log-config-filename LOG_CONFIG_FILENAME]
                     [--tracing {function}] [-? ?]
                     [-P OUTPUT_FILENAME_PATTERN] [-F] [-I STDIN] [-L STDLOG]
                     [-E STDERR] [-S STDOUT]
gff2histogram: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'gff2plot'
```

gff2stats.py - count features, etc. in gff file

Tags Genomics Intervals GFF GTF Summary

Purpose

This script generates summary statistics over features, source, gene_id and transcript_id in one or more *gff* or *gtf* formatted files.

Usage

Input is either a gff or gtf file; gtf input must be specified with the –is-gtf option.

Example:

```
python gff2stats.py --is-gtf example.gtf > example_sum.tsv

cat example.gtf

19  processed_transcript  exon  6634666509  .  -  .  gene_id "ENSG00000225373";␣
↪transcript_id "ENST00000592209" ...
19  processed_transcript  exon  6052160747  .  -  .  gene_id "ENSG00000225373";␣
↪transcript_id "ENST00000592209" ...
19  processed_transcript  exon  6010560162  .  -  .  gene_id "ENSG00000225373";␣
↪transcript_id "ENST00000592209" ...
19  processed_transcript  exon  6634666416  .  -  .  gene_id "ENSG00000225373";␣
↪transcript_id "ENST00000589741" ...

cat example_sum.tsv

track  contigs  strands  features  sources  genes  transcripts ...
stdin  1        2        4         23       2924   12752       ...
```

The counter used is dependent on the file type. For a gff file, the implemented counters are:

1. number of intervals per contig, strand, feature and source

For a gtf file, the additional implemented counters are:

1. number of genes, transcripts, single exon transcripts
2. summary statistics for exon numbers, exon sizes, intron sizes and transcript sizes

The output is a tab-separated table.

Options

The default action of `gff2stats` is to count over contigs, strand, feature and source. This assumes the input file is a gff file

There is a single option for this script:

```
``--is-gtf``
```

> The input file is gtf format. The output will therefore contain summaries over exon numbers, exon sizes, intron sizes and transcript sizes in addition to the the number of genes, transcripts and single exon transcripts.

Type:

```
python gff2stats.py --help
```

for command line help.

Command line options

```
usage: gff2stats [-h] [--version] [--is-gtf] [--timeit TIMEIT_FILE]
                 [--timeit-name TIMEIT_NAME] [--timeit-header]
                 [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                 [--log-config-filename LOG_CONFIG_FILENAME]
                 [--tracing {function}] [-? ?] [-P OUTPUT_FILENAME_PATTERN]
                 [-F] [-I STDIN] [-L STDLOG] [-E STDERR] [-S STDOUT]
gff2stats: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'gff2view'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'gff_decorate'
```

gtf2gff.py - convert a transcript set to genomic features

Tags Genomics Genesets Intervals Transformation GTF GFF

Purpose

This scripts converts a transcript set in a *gtf* formatted file into a set of features in a *gff* formatted file.

In other words, a gene set (gtf), which constitutes a hierarchical set of annotations, will be converted into a non-hierarchical list of genomic segments.

Various methods can be used to do the conversion (see command line argument `--method`):

exons annotate exons. Exonic segments are classified according to the transcript structure.

genome/full annotate genome with gene set. Genomic segments are labeled `intronic`, `intergenic`, etc. This annotation aggregates the information of multiple genes such that each annotation is either valid or ambiguous.

genes annotate genome using the information on a gene-by-gene basis. Multiple overlapping annotations will be created for each transcript. Redundant annotations will be merged.

great-domains regulatory domains using the basal+extended model according to GREAT.

promotors declare promoter regions. These segments might be overlapping. A promotor is the region x kb upstream of a transcription start site. The option `--promotor-size` sets the region width.

regulons declare regulatory regions. Regulatory regions contain the region x kb of upstream and downstream of a transciption start site. The options `--upstream-extension` and `-downstream` set the region width.

tts-regulons declare tts regulatory regions. tts-regulatory regions contain the region x kb of upstream and downstream of a transciption termination site. The options `--upstream-extension` and `-downstream` set the region width.

territories build gene territories around full length genes.

tss-territories build gene territories around transcription start sites.

In a simple setting, assume we have the two genes below, the first with a single transcript on the positive strand, the second on the negative strand:

```
       Gene A                    Gene B
        |---|                  |---|   |---|
     >>>>   >>>>            <<<<   <<<<   <<<<

Genome (simplified result without UTRs and flanks)

       exon   exon          exon   exon   exon
..---><--><-><--><---------<--><-><--><-><--><-----...
intergenic intron  intergenic  intron intron intergenic

Territories

     Gene A                    Gene B
<---------------------><------------------------------>

TSS-Territories

     Gene A                    Gene B
<--------->            <------------>
```

(continues on next page)

(continued from previous page)

```
Promotors

<----->               <----->
```

Genome

If `--method=genome`, the gene set is used to annotate the complete genome.

Note: The gtf file has to be sorted first by contig and then by position.

A segment in the genome will either be covered by:

cds a coding exon (also: CDS, start_codon).

utr a UTR (also: stop_codon)

5flank, 3flank, flank an upstream/downstream segment of defined size. If the intergenic region is too small to accomodate a flank, the regions is just 'flank'.

intergenic intergenic region.

5telomeric, 3telomeric telomeric region (before/after first/last gene).

intronic intronic region. An intron has a minimum size of 30 bases.

frameshift frameshift. Introns of less than 4 residues length

ambiguous in case of overlapping genes, regions are designated ambiguous

unknown unknown are `intronic` regions that are less than the minimum size of an intron (default: 30) and larger than the size of frameshift (default:4). These could be either genuine small introns or they could be artefactual arising from collapsing the exons within a gene model.

All segments are annotated by their closest gene. Intergenic regions are annotated with their two neighbouring genes. The upstream gene is listed in the attribute gene_id, the downstream one is listed in the attribute downstream_gene_id.

Genes

If `--method=genes`, the gene set is used to annotate the complete genome.

Note: The gtf file has to be sorted by gene.

A segment in the genome will be annotated as:

cds a coding exon

utr5, utr3 a 5' or 3' utr

exon an exon. Exons are further classified into first, middle and last exons.

intronic an intronic region. Intronic regions are further divided into first, middle, last.

upstream, downstream upstream/downstream regions in 5 intervals of a total of 1kb (see option –flank-size to increase the total size).

Territories

If `--method=territories`, the gene set is used to define gene territories. Territories are segments around genes and are non-overlapping. Exons in a gene are merged and the resulting the region is enlarged by –radius. Overlapping territories are divided at the midpoint between the two genes. The maximum extent of a territory is limited by the option `--territory-extension`

Note: The gtf file has to be sorted first by contig and then by position.

Note: Genes should already have been merged (gtf2gtf –merge-transcripts)

TSSTerritories

If `--method=tss-territories`, the gene set is used to define gene territories. Instead of the full gene length as in *Territories*, only the tss is used to define a territory. Territories are segments around genes and are non-overlapping. Overlapping territories are divided at the midpoint between the two genes. The maximum extent of a territory is limited by the option `--territory-extension`.

Note: The gtf file has to be sorted first by contig and then by position.

Note: Genes should already have been merged (gtf2gtf –merge-transcripts)

The domain definitions corresponds to the `nearest gene` rule in GREAT.

GREAT-Domains

Define GREAT regulatory domains. Each TSS in a gene is associated with a basal region. The basal region is then extended upstream to the basal region of the closest gene, but at most to –radius. In the case of overlapping genes, the extension is towards the next non-overlapping gene.

This is the "basal plus extension" rule in GREAT. Commonly used are 5+1 with 1 Mb extension. To achieve this, use for example:

```
cgat gtf2gff    --genome-file=hg19    --method=great-domains    --upstream-
↪extension=5000    --downstream-extension=1000    --territory-extension=1000000    <␣
↪in.gtf > out.gff
```

If there are a multiple TSS in a transcript, the basal region extends from the first to the last TSS plus the upstream/downstream flank.

Exons

If `--method=exons`, exons are annotated by their dispensibility.

Note: The gtf file should be sorted by genes

For each exon, the following additional fields are added to the gtf file:

ntranscripts number of transcripts

nused number of transcripts using this exon

positions positions of exon within transcripts. This is a `,` separated list of tuples `pos:total`. For example, `1:10,5:8` indicates an exon that appears in first position in a ten exon transcript and fifth position in an eight exon transcript. The position is according to the direction of transcription.

Note: overlapping but non-identical exons, for example due to internal splice sites, are listed as separate exons. Thus the output is not fully flat as some segments could be overlapping (see output variable `noverlapping` in the log file).

The following example uses an ENSEMBL gene set:: (needs genome-file to run)

> gunzip < Mus_musculus.NCBIM37.55.gtf.gz | awk '$3 == "CDS"' | python gtf2gff.py –method=exons –restrict-source=protein_coding

Promoters

If `--method=promotors`, putative promotor regions are output. A promoter is a pre-defined segment upstream of the transcription start site. As the actual start site is usually not known, the start of the first exon within a transcript is used as a proxy. A gene can have several promotors associated with it, but overlapping promotor regions of the same gene will be merged. A promoter can extend into an adjacent upstream gene.

The `--restrict-source` option determines which GTF entries are output. The default is to output all entries but the user can choose from protein_coding, pseudogene or lncRNA.

The size of the promotor region can be specified by the command line argument `--promotor-size`.

Regulons

If `--method=regulons`, putative regulon regions are output. This is similar to a `promotor`, but the region extends both upstream and downstream from the transcription start site.

The `--restrict-source` option determines which GTF entries are output. The default is to output all entries but the user can choose from protein_coding, pseudogene or lncRNA.

The size of the promotor region can be specified by the command line argument `--upstream-extension` and `--downstream-extension`

If `--method=tts-regulons`, regulons will be defined around the transcription termination site.

Usage

Type:

```
cgat gtf2gff --method=genome --genome-file=hg19 < geneset.gtf > annotations.gff
```

For command line help:

```
cgat gtf2gff --help
```

Command line options

```
usage: gtf2gff [-h] [--version] [-g GENOME_FILE] [-i]
               [-s {protein_coding,pseudogene,lncRNA}]
               [-m {full,genome,exons,promotors,tts,regulons,tts-regulons,genes,
↪territories,tss-territories,great-domains}]
               [-r RADIUS] [-f FLANK] [--flank-increment-size INCREMENT]
               [-p PROMOTOR] [-u UPSTREAM] [-d DOWNSTREAM]
               [--gene-detail {introns+exons,exons,introns}]
               [--merge-overlapping-promotors]
               [--min-intron-length MIN_INTRON_LENGTH] [--is-unsorted]
               [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
               [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
               [--log-config-filename LOG_CONFIG_FILENAME]
               [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
               [-E STDERR] [-S STDOUT]
gtf2gff: error: argument -?: expected one argument
```

gtf2gtf.py - manipulate transcript models

Tags Genomics Genesets GTF Manipulation

Purpose

This script reads a gene set in *gtf* format from stdin, applies some transformation, and outputs a new gene set in *gtf* format to stdout. The transformation is chosen by the `--method` command line option.

Transformations available for use in this script can broadly be classified into four categories:

1. sorting gene sets
2. manipulating gene models
3. filtering gene sets
4. setting/resetting fields within a gtf file

Further options for working with gtf files are available in gff2gff.py, which can be run with the specification –is-gtf

Sorting gene sets

`sort`

Sorts entries in gtf file by one or more fields

option gene	order in which fields are sorted gene_id, contig, start
gene+transcript	gene_id, transcript_id, contig, start
contig+gene	contig, gene_id, transcript_id, start
transcript	transcript_id, contig, start
position	contig, start
position+gene	contig(gene_id, start)
gene+position	gene_id, contig, start
gene+exon	gene_id, exon_id

N.B. position+gene sorts by gene_id, start, then subsequently sorts flattened gene lists by contig, start

Manipulating gene-models

Options that can be used to alter the features represented in a *gtf* file. Only one method can be specified at once.

Input gtfs need to be sorted so that features for a gene or transcript appear consecutively within the file. This can be achevied using `--method=sort`.

genes-to-unique-chunks` Divide the complete length of a gene up into chunks that represent ranges of bases that are all present in the same set of transcripts. E.g. for two overlapping exons an entry will be output representing the overlap and a seperate entry each for the sequences only present in one. Ranges which are between the first TSS and last TTS but not present in any transcript (i.e. merged introns) are also output. Useful for DEXSeq like splicing analysis

find-retained-introns Finds intervals within a transcript that represent retained-introns, here a retained intron is considered to be an intron in one transcript that is entirely contianed within the exon of another. The retained intron will be assigned to the transcript with the containing exon. Where multiple, overlapping introns are contained within a single exon of a transcript, the union of the introns will be output. Thus when considering an indevidual transcript, outputs will be non-overlapping. However, overlapping, or even identical feature can be output if they belong to different transcripts.

merge-exons Merges overlapping exons for all transcripts of a gene, outputting the merged exons. Can be used in conjunction with `merge-exons-distance` to set the minimum distance that may appear between two exons before they are merged.If `--mark-utr` is set, the UTR regions will be output separately.

merge-transcripts Merges all transcripts of a gene. Outputs contains a single interval that spans the original gene (both introns and exons). If `--with-utr` is set, the output interval will also contain UTR.

`merge-genes`

Merges genes that have overlapping exons, outputting a single gene_id and transcript_id for all exons of overlapping genes. The input needs te sorted by transcript " (Does not merge intervals on different strands).

join-exons Joins together all exons of a transcript, outputting a single interval that spans the original transcript (both introns and exons). Input needs to be sorted by transcript.

intersect-transcripts Finds regions representing the intersection of all transcripts of a gene. Output will contain intervals spanning only those bases covered by all transcripts. If `--with-utr` is set, the UTR will also be included in the intersect. This method only uses `exon` or `CDS` features.

merge-introns Outputs a single interval that spans the region between the start of the first intron and the end of last intron. Single exons genes will not be output. The input needs to be sorted by gene

exons2introns Merges overlapping introns for all transcripts of a gene, outputting the merged introns. Use `--intron-min-length` to ignore merged introns below a specified length. Use `--intron-border` to specify a number of residues to remove at either end of output introns (residues are removed prior to filtering on size when used in conjunction with `--intron-min-length`).

transcripts2genes Cluster transcripts into genes by exon overlap ignoring any gene_ids in the *gtf* file. May be used in conjunction with `reset-strand`

The option `permit-duplicates` may be specified in order to allow gene-ids to be duplicated within the input *gtf* file (i.e. for the same gene-id to appear non-consecutively within the input file). However, this option currently only works for `merge-exons`, `merge-transcripts`, `merge-introns`, and `intersect-transcripts`. It DOES NOT work for `merge-genes`, `join-exons`, or `exons-file2introns`.

Filtering gene sets

Options that can be used to filter *gtf* files. For further detail see command line options.

Input gtfs need to be sorted so that features for a gene or transcript appear consecutively within the file. This can be achevied using `--method=sort --sort-order`.

filter When filtering on the basis of 'gene-id' or 'transcript-id' a filename containing ids to be removed may provided using `--map-tsv-file`. Alternatively, a random subsample of genes/transcripts may be retained using `--sam-fileple-size`. Use `--min-exons-length` in conjunction with `--sam-fileple-size` to specify a minimum length for genes/transcripts to be retained. Use `--ignore-strand` to set strand to '.' in output.

Other filter options include longest-gene, longest-transcript, or representative-transcript.

When filtering on the basis of gene-id, transcript-id or longest-gene, `--invert-filter` may be used to invert the selection.

remove-overlapping Given a second *gff* formatted file (`--file-gff`) removes any features overlapping. Any transcripts that intersect intervals in the supplied file are removed. (Does not account for strand.)

remove-duplicates Remove duplicate features from *gtf* file. The type of feature to be removed is set by the option `-duplicate-feature`. Setting `--duplicate-feature` to 'gene', 'transcript', or 'coordinates' will remove any interval for which non-consecutive occurrances of specified term appear in input *gtf* file. Setting to 'ucsc', will remove any interval for which transcript-id contains '_dup'.

Setting fields

Options for altering fields within *gtf*.

rename-genes With a mapping file is provided using `--map-tsv-file`, renames the gene_id to the one supplied. Outputs a *gtf* file with field renamed. Any entry in input *gtf* not appearing in mapping file is discarded.

rename-transcripts as `rename-genes`, but renames the transcript_id.

add-protein-id Takes a map of transcript_id to protein_id from the a tsv file (see option `--map-tsv-file`) and appends the protein_id provided to the attributes field. Any entry with a transcript_id not appearing in the tsv file is discarded.

renumber-genes Renumber genes from 1 using the pattern provided in `--pattern-identifier`.

renumber-transcripts Renumber transcripts from 1 using the pattern provided in `--pattern-identifier`.

unset-genes Renumber genes from 1 using the pattern provided in `--pattern-identifier`. Transcripts with the same gene-id in the input *gtf* file will have different gene-ids in the output *gtf* file.

set-transcript-to-gene Will set the transcript-id to the gene-id for each feature.

set-gene-to-transcript Will set the gene-id to the transcript-id for each each feature.

set-protein-to-transcript Will append transcript_id to attributes field as 'protein_id'

set-score-to-distance Will reset the score field (field 6) of each feature in input *gtf* to be the distance from transcription start site to the start of the feature. (Assumes input file is sorted by transcript-id)

set-gene_biotype-to-source Sets the `gene_biotype` attribute from the source column. Will only set if biotype attribute is not present in the current record.

rename-duplicates Rename duplicate gene_ids and transcript_ids by addition of numerical suffix

set-source-to-transcript_biotype Sets the source attribute to the `transcript_biotype` attribute. Will only set if `transcript_biotype` attribute is present in the current record.

Usage

The following example sorts the input gene set by gene (`method=sort`) so that it can be used as input for `method=intersect-transcripts` that outputs genomic the genomic regions within a gene that is covered by all transcripts in a gene. Finally, the resultant transcripts are renamed with the pattern "MERGED_%i":

```
cgat gtf2gtf
        --method=sort
        --sort-order=gene     | cgat gtf2gtf
           --method=intersect-transcripts
           --with-utr
| cgat gtf2gtf
           --method=renumber-transcripts
           --pattern-identifier=MERGED_%i
```

Type:

```
cgat gtf2gtf --help
```

for command line options.

Command line Options

```
usage: gtf2gtf [-h] [--version] [--merge-exons-distance MERGE_EXONS_DISTANCE]
               [--pattern-identifier PATTERN]
               [--sort-order {gene,gene+transcript,transcript,position,contig+gene,
↪position+gene,gene+position,gene+exon}]
               [--mark-utr] [--without-utr]
               [--filter-method {gene,transcript,longest-gene,longest-transcript,
↪representative-transcript,proteincoding,lincrna}]
               [-a tsv] [--gff-file GFF] [--invert-filter]
               [--sample-size SAMPLE_SIZE]
               [--intron-min-length INTRON_MIN_LENGTH]
               [--min-exons-length MIN_EXONS_LENGTH]
               [--intron-border INTRON_BORDER] [--ignore-strand]
               [--permit-duplicates]
               [--duplicate-feature {gene,transcript,both,ucsc,coordinates}]
```

(continues on next page)

(continued from previous page)

```
               [--use-gene-id]
               [-m {add-protein-id,exons2introns,filter,find-retained-introns,genes-
↪to-unique-chunks,intersect-transcripts,join-exons,merge-exons,merge-transcripts,
↪merge-genes,merge-introns,remove-overlapping,remove-duplicates,rename-genes,rename-
↪transcripts,rename-duplicates,renumber-genes,renumber-transcripts,set-transcript-to-
↪gene,set-gene-to-transcript,set-protein-to-transcript,set-score-to-distance,set-
↪gene_biotype-to-source,set-source-to-transcript_biotype,sort,transcript2genes,unset-
↪genes}]
               [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
               [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
               [--log-config-filename LOG_CONFIG_FILENAME]
               [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
               [-E STDERR] [-S STDOUT]
gtf2gtf: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'gtf2reads'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 129, in main
    module = imp.load_module(command, file, pathname, description)
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 235, in load_module
    return load_source(name, filename, file)
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 172, in load_source
    module = _load(spec)
  File "<frozen importlib._bootstrap>", line 696, in _load
  File "<frozen importlib._bootstrap>", line 677, in _load_unlocked
  File "<frozen importlib._bootstrap_external>", line 728, in exec_module
  File "<frozen importlib._bootstrap>", line 219, in _call_with_frames_removed
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/tools/gtf2table.py", line 467, in <module>
    import cgat.GeneModelAnalysis as GeneModelAnalysis
  File "cgat/GeneModelAnalysis.pyx", line 30, in init cgat.GeneModelAnalysis
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/SequenceProperties.py", line 49, in <module>
    import Bio.Alphabet.IUPAC
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/Bio/Alphabet/__init__.py", line 21, in <module>
    "Bio.Alphabet has been removed from Biopython. In many cases, the alphabet can␣
↪simply be ignored and removed from scripts. In a few cases, you may need to specify␣
↪the ``molecule_type`` as an annotation on a SeqRecord for your script to work␣
↪correctly. Please see https://biopython.org/wiki/Alphabet for more information."
```

(continues on next page)

(continued from previous page)

```
ImportError: Bio.Alphabet has been removed from Biopython. In many cases, the␣
↪alphabet can simply be ignored and removed from scripts. In a few cases, you may␣
↪need to specify the ``molecule_type`` as an annotation on a SeqRecord for your␣
↪script to work correctly. Please see https://biopython.org/wiki/Alphabet for more␣
↪information.
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'gtfs2graph'
```

gtf2fasta.py - annotate genomic bases from a gene set

Tags Genomics Genesets Sequences GTF FASTA Transformation

Purpose

This script can be used for a quick-and-dirty annotation of variants in a genome. It is most appropriately used in exploratory analyses of the effect of variants/alleles.

For a better prediction of variant effects in coding sequences, see <*no title*> and <*no title*>.

If you wish to convert gtf intervals into fasta sequences, use gff2fasta.py.

This script takes a *gtf* formatted file from ENSEMBL and annotates each base in the genome according to its *function*. The script multiplexes both strands with lower- case characters referring to the forward strand and upper-case characters referring to the reverse strand.

The codes and their meaning are:

code	description
a	first codon position within a complete codon
b	second codon position within a complete codon
c	third codon position within a complete codon
d	coding base, but in multiple frames or strands
e	non-coding base in exon
f	frame-shifted base
g	intergenic base
i	intronic base
l	base in other RNA
m	base in miRNA
n	base in snRNA
o	base in snoRNA
r	base in rRNA (both genomic and mitochondrial)
p	base in pseudogene (including transcribed, unprocessed and processed)
q	base in retrotransposon
s	base within a splice signal (GT/AG)
t	base in tRNA (both genomic and mitochondrial)
u	base in 5' UTR
v	base in 3' UTR
x	ambiguous base with multiple functions.
y	unknown base

Output files

The annotated genome is output on stdout.

The script creates the following additional output files:

counts Counts for each annotations

junctions Splice junctions. This is a tab separated table linking residues that are joined via features. The coordinates are forward/reverse coordinates.

The columns are:

contig the contig

strand direction of linkage

end last base of exon in direction of strand

start first base of exon in direction of strand

frame frame base at second coordinate (for coding sequences)

Known problems

The stop-codon is part of the UTR. This has the following effects:

- On the mitochondrial chromosome, the stop-codon might be used for ncRNA transcripts and thus the base is recorded as ambiguous.
- On the mitochondrial chromosome, alternative transcripts might read through a stop-codon (RNA editing). The codon itself will be recorded as ambiguous.

Usage

For example:

```
zcat hg19.gtf.gz | python gtf2fasta.py --genome-file=hg19 > hg19.annotated
```

Type:

```
python gtf2fasta.py --help
```

for command line help.

Command line options

--genome-file required option. filename for genome fasta file

--ignore-missing transcripts on contigs not in the genome file will be ignored

--min-intron-length intronic bases in introns less than specified length will be marked "unknown"

```
usage: gtf2fasta [-h] [--version] [-g GENOME_FILE] [-i]
                 [--min-intron-length MIN_INTRON_LENGTH] [-m {full}]
                 [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                 [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                 [--log-config-filename LOG_CONFIG_FILENAME]
                 [--tracing {function}] [-? ?] [-P OUTPUT_FILENAME_PATTERN]
                 [-F] [-I STDIN] [-L STDLOG] [-E STDERR] [-S STDOUT]
gtf2fasta: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'snp2table'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
```

(continues on next page)

(continued from previous page)

```
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'psl2fasta'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'psl2stats'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'psl2table'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'maq2assembly'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'maq2psl'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'analyze_readpositions'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'combine_gff'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'quality2fasta'
```

bam2wiggle.py - convert bam to wig/bigwig file

Tags Genomics NGS Intervals Conversion BAM WIGGLE BIGWIG BEDGRAPH

Purpose

convert a bam file to a bigwig or bedgraph file.

Depending on options chosen, this script either computes the densities itself or makes use of faster solutions if possible. The script requires the executables `wigToBigWig` and `bedToBigBed` to be in the user's PATH.

If no –shift-size or –extend option are given, the coverage is computed directly on reads. Counting can be performed at a certain resolution.

The counting currently is not aware of spliced reads, i.e., an inserted intron will be included in the coverage.

If –shift-size or –extend are given, the coverage is computed by shifting read alignment positions upstream for positive strand reads or downstream for negative strand reads and extend them by a fixed amount.

For RNASEQ data it might be best to run genomeCoverageBed directly on the bam file.

Usage

Type:

```
cgat bam2wiggle           --output-format=bigwig           --output-filename-
↪pattern=out.bigwig in.bam
```

to convert the *bam* file file:*in.bam* to *bigwig* format and save the result in `out.bigwig`.

Command line options

```
usage: bam2wiggle [-h] [--version] [-o {bedgraph,wiggle,bigbed,bigwig,bed}]
                  [-s SHIFT] [-e EXTEND] [-p SPAN] [-m]
                  [--scale-base SCALE_BASE] [--scale-method {none,reads}]
                  [--max-insert-size MAX_INSERT_SIZE]
                  [--min-insert-size MIN_INSERT_SIZE] [--timeit TIMEIT_FILE]
                  [--timeit-name TIMEIT_NAME] [--timeit-header]
                  [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                  [--log-config-filename LOG_CONFIG_FILENAME]
                  [--tracing {function}] [-? ?] [-P OUTPUT_FILENAME_PATTERN]
                  [-F] [-I STDIN] [-L STDLOG] [-E STDERR] [-S STDOUT]
bam2wiggle: error: argument -?: expected one argument
```

bed2annotator.py - convert bed to annotator format

Tags Python

Purpose

This script converts a bed file into annotator compatible regions. Depending on the option –section this script will create:

segments a segments file

annotations a file with annotations. Each bed track is a separate annotation.

workspace a file with a workspace

Usage

Example:

```
python bed2annotator2tsv.py --help
```

Type:

```
python bed2annotator2tsv.py --help
```

for command line help.

Command line options

```
usage: bed2annotator [-h] [-g GENOME_FILE] [-f FEATURES] [-i FILES]
                     [-a ANNOTATIONS] [--map-tsv-file INPUT_FILENAME_MAP]
                     [-l MAX_LENGTH] [-m]
                     [-s {segments,annotations,workspace}] [--subset SUBSETS]
                     [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                     [--timeit-header] [--random-seed RANDOM_SEED]
                     [-v LOGLEVEL] [--log-config-filename LOG_CONFIG_FILENAME]
                     [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                     [-E STDERR] [-S STDOUT]
bed2annotator: error: argument -?: expected one argument
```

bed2graph.py - compute the overlap graph between two bed files

Tags Python

Purpose

This script ouputs a list of the names of all overlapping intervals between two bed files.

Usage

Type:

```
python bed2graph.py A.bed.gz B.bed.gz > graph.out
```

for command line help.

Command line options

```
usage: bed2graph [-h] [--version] [-o {full,name}] [--timeit TIMEIT_FILE]
                 [--timeit-name TIMEIT_NAME] [--timeit-header]
                 [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                 [--log-config-filename LOG_CONFIG_FILENAME]
                 [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                 [-E STDERR] [-S STDOUT]
bed2graph: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'bed2psl'
```

chain2psl.py - convert a chain file to a psl file

Tags Genomics Intervals GenomeAlignment PSL CHAIN Conversion

Purpose

convert a UCSC chain formatted file to a UCSC psl formatted file.

This tool is equivalent to the UCSC tool chainToPsl except that it will not compute the number of matching, mismatching, etc. bases and thus does not require the sequences.

The nomenclature the UCSC uses for its chain files is `targetToQuery.chain` for mapping `query` to `target` (reference). According to the UCSC documentation, `target` is the first entry in `chain` files.

We have been using the nomenclature `QueryToTarget.psl`. In following this convention, the correct way to converting a psl file is:

```
python chain2psl.py < targetToQuery.chain > QueryToTarget.psl
```

If you would like to keep the TargetToQuery convention, you will need to add a pslSwap:

```
python chain2psl.py < targetToQuery.chain | pslSwap stdin stdout > targetToQuery.psl
```

Usage

For example:

```
cgat chain2psl.py < in.chain > out.psl
```

Type:

```
cgat chain2psl.py --help
```

for command line help.

Command line options

```
usage: chain2psl [-h] [--version] [--timeit TIMEIT_FILE]
                 [--timeit-name TIMEIT_NAME] [--timeit-header]
                 [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                 [--log-config-filename LOG_CONFIG_FILENAME]
                 [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                 [-E STDERR] [-S STDOUT]
chain2psl: error: argument -?: expected one argument
```

diff_bed.py - count differences between several bed files

Tags Genomics Intervals BED Comparison

Purpose

Compute overlap statistics between multiple bed files. For each pairwise comparison, this script outputs the number of intervals (exons) and bases overlapping.

Using the `--update` option, a table can be incrementally updated with additional comparisons.

The strand of intervals is ignored in comparisons.

Column	*Content*
set	Name of the set
nexons_total	number of intervals in set
nexons_ovl	number of intervals overlapping
nexons_unique	number of unique intervals
nbases_total	number of bases in gene set
nbases_ovl	number of bases overlapping
nbases_unique	number of unique bases

Usage

For example:

```
python diff_bed.py *.bed.gz > out.tsv
```

To update results from a previous run, type:

```
python diff_bed.py --update=out.tsv *.bed.gz > new.tsv
```

Type:

```
python diff_bed.py --help
```

for command line help.

Command line options

```
usage: diff-bed [-h] [--version] [-u FILENAME_UPDATE] [-p PATTERN_ID] [-t]
                [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                [--log-config-filename LOG_CONFIG_FILENAME]
                [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                [-E STDERR] [-S STDOUT]
diff-bed: error: argument -?: expected one argument
```

fasta2bed.py - segment sequences

Tags Genomics Sequences Intervals FASTA BED Conversion

Purpose

This script takes a genomic sequence in *fasta* format and applies various segmentation algorithms.

The methods implemented (`--methods`) are:

cpg output all locations of cpg in the genome

fixed-width-windows-gc output fixed width windows of a certain size adding their G+C content as score

gaps ouput all locations of assembly gaps (blocks of *N*) in the genomic sequences

ungapped output ungapped locations in the genomic sequences

Usage

Type:

```
python fasta2bed.py --method=gap < in.fasta > out.bed
```

Type:

```
python fasta2bed.py --help
```

for command line help.

Command line options

```
usage: fasta2bed [-h] [--version]
                 [-m {fixed-width-windows-gc,cpg,windows-cpg,gaps,ungapped,windows}]
                 [-w WINDOW_SIZE] [-s WINDOW_SHIFT] [--min-cpg MIN_CPG]
                 [--min-interval-length MIN_LENGTH] [--timeit TIMEIT_FILE]
                 [--timeit-name TIMEIT_NAME] [--timeit-header]
                 [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                 [--log-config-filename LOG_CONFIG_FILENAME]
                 [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                 [-E STDERR] [-S STDOUT]
fasta2bed: error: argument -?: expected one argument
```

gff2table.py - compute features for intersection of two gff files

Tags Genomics Intervals Annotation Comparison GFF

Purpose

collect intervals from two gff files and compute features based on their intersection. The script is intended to compute properties for a set of non-overlapping windows.

Transforms:

- none: no transform
- overlap: overlap between set1 and set2
- complement: part of set1 that is not covered by set2
- **third_codon: only takes every third position. Needs frame information** in the gff file.

Decorators:

- GC: G+C content of intervals
- count: number of windows
- mean-length: mean length of intervals overlapping with window

Usage

Example:

```
python gff2table.py --help
```

Type:

```
python gff2table.py --help
```

for command line help.

Command line options

```
usage: gff2table [-h] [--version] [-g GENOME_FILE] [-w FILENAME_WINDOWS]
                 [-d FILENAME_DATA] [--is-gtf] [-f {GC}]
                 [-c {counts,gc,gc3,mean-length,median-length,percent-coverage,median-
↪score,mean-score,stddev-score,min-score,max-score}]
                 [-e] [-t {none,overlap,complement,third_codon}]
                 [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                 [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                 [--log-config-filename LOG_CONFIG_FILENAME]
                 [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                 [-E STDERR] [-S STDOUT]
gff2table: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
```

(continues on next page)

(continued from previous page)

```
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'index2gff'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'maf2psl'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'snp2counts'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'snp2maf'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'psl2chain'
```

Visualization

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'plot_data'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'plot_histogram'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'plot_matrix'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'png2svg'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
```

(continues on next page)

(continued from previous page)

```
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'go2svg'
```

Sequences and rates

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'mask_fasta'
```

index_fasta.py - Index fasta formatted files

Tags Genomics Sequences FASTA Manipulation

Purpose

This script indexes one or more *fasta* formatted files into a database that can be used by other scripts in the cgat code collection and `IndexedFasta` for quick access to a particular part of a sequence. This is very useful for large genomic sequences.

By default, the database is itself a *fasta* formatted file in which all line breaks and other white space characters have been removed. Compression methods are available to conserve disk space, though they do come at a performance penalty.

The script implements several indexing and compression methods. The default method uses no compression and builds a simple random access index based on a table of absolute file positions. The sequence is stored in a plain fasta file with one line per sequence allowing to extract a sequence segment by direct file positioning.

Alternatively, the sequence can be block-compressed using different compression methods (gzip, lzo, bzip). These are mostly for research purposes.

See also http://pypi.python.org/pypi/pyfasta for another implementation. Samtools provides similar functionality with the `samtools faidx` command and block compression has been implemented in the **`bgzip http://samtools.sourceforge.net/tabix.shtml>`_** tool.

The script permits supplying synonyms to the database index. For example, setting `--synonyms=chrM=chrMT` will ensure that the mitochondrial genome sequence is returned both for the keys `chrM` and `chrMT`.

Examples

Index a collection of fasta files in a compressed archive:

```
python index_fasta.py oa_ornAna1_softmasked ornAna1.fa.gz > oa_ornAna1_softmasked.log
```

To retrieve a segment:

```
python index_fasta.py --extract=chr5:1000:2000 oa_ornAna1_softmasked
```

Indexing from a tar file is possible:

```
python index_fasta.py oa_ornAna1_softmasked ornAna1.tar.gz > oa_ornAna1_softmasked.log
```

Indexing from stdin requires to use the - place-holder:

```
zcat ornAna1.fa.gz | python index_fasta.py oa_ornAna1_softmasked - > oa_ornAna1_
↪softmasked.log
```

Usage

Type:

```
cgat index_genome DATABASE [SOURCE...|-] [OPTIONS]
cgat index_genome DATABASE [SOURCE...|-] --compression=COMPRESSION --random-access-
↪points=100000
```

To create indexed DATABASE from SOURCE. Supply - as SOURCE to read from stdin. If the output is to be compressed, a spacing for the random access points must be supplied.

Type:

```
cgat index_genome DATABASE --extract=CONTIG:[STRAND]:START:END
```

To extract the bases on the STRAND strand, between START to END from entry CONTIG, from DATABASE.

Command line options

```
usage: index-fasta [-h] [--version] [-e EXTRACT]
                   [-i {one-forward-open,zero-both-open}] [-s SYNONYMS] [-b]
                   [--benchmark-num-iterations BENCHMARK_NUM_ITERATIONS]
                   [--benchmark-fragment-size BENCHMARK_FRAGMENT_SIZE]
                   [--verify VERIFY]
                   [--verify-iterations VERIFY_NUM_ITERATIONS]
                   [--file-format {fasta,auto,fasta.gz,tar,tar.gz}] [-a]
                   [--allow-duplicates] [--regex-identifier REGEX_IDENTIFIER]
                   [--force-output] [-t {solexa,phred,bytes,range200}]
                   [-c {lzo,zlib,gzip,dictzip,bzip2,debug}]
                   [--random-access-points RANDOM_ACCESS_POINTS]
                   [--compress-index] [--timeit TIMEIT_FILE]
                   [--timeit-name TIMEIT_NAME] [--timeit-header]
                   [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                   [--log-config-filename LOG_CONFIG_FILENAME]
                   [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
```

(continues on next page)

(continued from previous page)

```
                    [-E STDERR] [-S STDOUT]
index-fasta: error: argument -?: expected one argument
```

diff_fasta.py - compare contents of two fasta files

Tags Genomics Sequences FASTA Comparison

Purpose

This script takes two sets of fasta sequences and matches the identifiers. It then compares the sequences with the same identifiers and, depending on the output options selected, outputs

- which sequences are missing
- which sequences are identical
- which sequences are prefixes/suffixes of each other

An explanatory field is appended to output sequence identifiers. An explanation of the different field values is provided in the log.

Options

-s, --correct-gap-shift This option will correct shifts in alignment gaps between two sequences being compared

-1, --pattern1 regular expression pattern to extract identifier from in sequence 1

-2, --pattern2 regular expression pattern to extract identifier from in sequence 2

Depending on the option `--output-section` the following are output:

diff identifiers of sequences that are different

seqdiff identifiers of sequences that are different plus sequence

missed identifiers of seqences that are missing from one set or the other

This script is of specialized interest and has been used in the past to check if ENSEMBL gene models had been correctly mapped into a database schema.

Usage

Example:

```
cat a.fasta | head

>ENSACAP00000004922
MRSRNQGGESSSSGKFSKSKPIINTGENQNLQEDAKKKNKSSRKEE ...
>ENSACAP00000005213
EEEEDESNNSYLYQPLNQDPDQGPAAVEETAPSTEPALDINERLQA ...
>ENSACAP00000018122
LIRSSSMFHIMKHGHYISRFGSKPGLKCIGMHENGIIFNNNPALWK ...
```

(continues on next page)

(continued from previous page)

```
python diff_fasta.py --output-section=missed --output-section=seqdiff a.fasta b.fasta

cat diff.out

# Legend:
# seqs1:          number of sequences in set 1
# seqs2:          number of sequences in set 2
# same:           number of identical sequences
# diff:           number of sequences with differences
# nmissed1:       sequences in set 1 that are not found in set 2
# nmissed2:       sequences in set 2 that are not found in set 1
# Type of sequence differences
# first:          only the first residue is different
# last:           only the last residue is different
# prefix:         one sequence is prefix of the other
# selenocysteine: difference due to selenocysteines
# masked:         difference due to masked residues
# fixed:          fixed differences
# other:          other differences
```

Type:

```
python diff_fasta.py --help
```

for command line help.

Command line options

```
usage: diff-fasta [-h] [--version] [-s] [-1 PATTERN1] [-2 PATTERN2]
                  [-o {diff,missed,seqdiff}] [--timeit TIMEIT_FILE]
                  [--timeit-name TIMEIT_NAME] [--timeit-header]
                  [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                  [--log-config-filename LOG_CONFIG_FILENAME]
                  [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                  [-E STDERR] [-S STDOUT]
diff-fasta: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'extractseq'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
```

(continues on next page)

(continued from previous page)

```
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'quality2masks'
```

Matrices and Tables

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'combine_histograms'
```

csvs2csv.py - join tables

Tags Python

Purpose

This script reads several tab-separated tables and joins them.

Note: working with multiple columns per table and sorting is not implemented correctly and likely to fail.

Usage

Example:

```
python combine_tables.py --help
```

Type:

```
python combine_tables.py --help
```

for command line help.

Command line options

```
usage: csvs2csv [-h] [--version] [-t] [-i] [-m MISSING_VALUE]
                [--header-names HEADERS] [-c COLUMNS] [-g GLOB] [-s SORT] [-e]
                [--sort-keys {numeric,alphabetic}] [--keep-empty]
                [--add-file-prefix] [--regex-filename REGEX_FILENAME]
                [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                [--log-config-filename LOG_CONFIG_FILENAME]
                [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                [-E STDERR] [-S STDOUT]
csvs2csv: error: argument -?: expected one argument
```

csv2csv.py - operate on tables

Tags Python

Purpose

operate on tables.

Usage

Example:

```
python csv2csv.py --help
```

Type:

```
python csv2csv.py --help
```

for command line help.

Command line options

```
usage: csv2csv [-h] [--version] [-s SORT] [--timeit TIMEIT_FILE]
               [--timeit-name TIMEIT_NAME] [--timeit-header]
               [--random-seed RANDOM_SEED] [-v LOGLEVEL]
               [--log-config-filename LOG_CONFIG_FILENAME]
               [--tracing {function}] [-? ?] [--csv-dialect CSV_DIALECT]
               [-I STDIN] [-L STDLOG] [-E STDERR] [-S STDOUT]
csv2csv: error: argument -?: expected one argument
```

csv2db.py - upload table to database

Tags Python

Purpose

create a table from a csv separated file and load data into it.

This module supports backends for postgres and sqlite3. Column types are auto-detected.

Read a table from stdin and create an sqlite3 database. By default, the database will reside in a file called csvdb and in a table csv.

Todo: Use file import where appropriate to speed up loading. Currently, this is not always the case.

Usage

Example:

```
python csv2db.py -b sqlite < stdin
```

Type:

```
python csv2db.py --help
```

for command line help.

Command line options

```
usage: csv2db [-h] [--version] [--csv-dialect DIALECT] [-m MAP] [-t TABLENAME]
              [-H HEADER_NAMES] [--replace-header] [-l]
              [--chunk-size CHUNK_SIZE] [--ignore-column IGNORE_COLUMNS]
              [--rename-column RENAME_COLUMNS] [--first-column FIRST_COLUMN]
              [-e] [-i INDICES] [-a] [--retry] [--append] [--utf8]
              [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
              [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
              [--log-config-filename LOG_CONFIG_FILENAME]
              [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG] [-E STDERR]
              [-S STDOUT] [--database-url DATABASE_URL]
              [--database-schema DATABASE_SCHEMA]
csv2db: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
```

(continues on next page)

(continued from previous page)

```
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'csv2xls'
```

csv_cut.py - select columns from a table

Tags Python

Purpose

extract named columns from a csv formatted table

Todo: describe purpose of the script.

Usage

Extract the two columns gene and length from a table in standard input:

```
python csv_cut.py gene length < stdin
```

The script permits the use of patterns. For example, the command will select the column gene and all columns that contain the part 'len':

```
python csv_cut.py gene %len% < stdin
```

Type:

```
python csv_cut.py --help
```

for command line help.

Command line options

```
usage: csv-cut [-h] [-r] [-u] [-l] [-f FILENAME_FIELDS] [--timeit TIMEIT_FILE]
               [--timeit-name TIMEIT_NAME] [--timeit-header]
               [--random-seed RANDOM_SEED] [-v LOGLEVEL]
               [--log-config-filename LOG_CONFIG_FILENAME]
               [--tracing {function}] [-? ?] [--csv-dialect CSV_DIALECT]
               [-I STDIN] [-L STDLOG] [-E STDERR] [-S STDOUT]
csv-cut: error: argument -?: expected one argument
```

csv_intersection.py - intersect two tables

Tags Python

Purpose

Todo: describe purpose of the script.

Usage

Example:

```
python csv_intersection.py --help
```

Type:

```
python csv_intersection.py --help
```

for command line help.

Command line options

```
usage: csv-intersection [-h] [--version] [-u] [--timeit TIMEIT_FILE]
                        [--timeit-name TIMEIT_NAME] [--timeit-header]
                        [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                        [--log-config-filename LOG_CONFIG_FILENAME]
                        [--tracing {function}] [-? ?]
                        [--csv-dialect CSV_DIALECT] [-I STDIN] [-L STDLOG]
                        [-E STDERR] [-S STDOUT]
csv-intersection: error: argument -?: expected one argument
```

csv_rename.py - rename columns in a table

Tags Python

Purpose

rename columns in a csv file

Usage

Example:

```
csv_rename.py gene=id < stdin
```

Type:

```
python csv_rename.py --help
```

for command line help.

Command line options

```
usage: csv-rename [-h] [--version] [-r] [-u] [-f FILENAME_FIELDS]
                  [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                  [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                  [--log-config-filename LOG_CONFIG_FILENAME]
                  [--tracing {function}] [-? ?] [--csv-dialect CSV_DIALECT]
                  [-I STDIN] [-L STDLOG] [-E STDERR] [-S STDOUT]
csv-rename: error: argument -?: expected one argument
```

csv_set.py - set operations on a table

Tags Python

Purpose

Todo: describe purpose of the script.

Usage

Example:

```
python csv_set.py --help
```

Type:

```
python csv_set.py --help
```

for command line help.

Command line options

```
usage: csv-set [-h] [--version] [-u] [-1 JOIN_FIELDS1] [-2 JOIN_FIELDS2]
               [-m {intersection,rest,union}] [--timeit TIMEIT_FILE]
               [--timeit-name TIMEIT_NAME] [--timeit-header]
               [--random-seed RANDOM_SEED] [-v LOGLEVEL]
               [--log-config-filename LOG_CONFIG_FILENAME]
               [--tracing {function}] [-? ?] [--csv-dialect CSV_DIALECT]
               [-I STDIN] [-L STDLOG] [-E STDERR] [-S STDOUT]
csv-set: error: argument -?: expected one argument
```

cat_tables.py - concatenate tables

Tags Python

Purpose

concatenate tables. Headers of subsequent files are ignored.

Usage

Type:

```
python <script_name>.py --help
```

for command line help.

Command line options

```
usage: cat-tables [-h] [--version] [--timeit TIMEIT_FILE]
                  [--timeit-name TIMEIT_NAME] [--timeit-header]
                  [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                  [--log-config-filename LOG_CONFIG_FILENAME]
                  [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                  [-E STDERR] [-S STDOUT]
cat-tables: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'matrix2matrix'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'matrix2stats'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'sparse2full'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'filter_tokens'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'table2graph'
```

table2table.py - operate on tables

Tags Python

Purpose

This script implements a few methods for manipulating tables.

Methods working on all tables:

transpose transpose a table

split-fields Split muliple-value fields in each row at `--separator`. Output multiple rows with all combinations.

group Group values by column

join-column Join rows in a table by columns

expand-table If a field in a row contains multiple values, the row is expanded into multiple rows such that all values have space.

flatten-table Output a table as row/column/value tuples.

as-column Output table as a single column. Colums in the original table are appended and output.

collapse-table Collapse a table of two columns with row names in the first column. Outputs a table with multiple columns for each row name.

Methods for numerical columns

Some methods make only sense for columns containing numerical values. If a table contains both numerical and non-numerical data, the numerical columns can be specified by the `--columns` option.

normalize-by-value divide all cells in a table by a value

multiply-by-value multiply all cells in a table by a value

lower-bound replace all cells with a value of less than lower bound with the lower bound.

upper-bound replace all cells with a value of more than upper bound with the upper bound.

normalize-by-table divide each cell in a table with the corresponding entry in a secondary table.

normalize-by-max divide table columns by maximum per column

kullback-leibler compute kullback-leibler divergence between two columns. Compute both D(a||b), D(b||a) and (D(a||b) + D(b||a)) / 2

rank substitute cells with their ranks in a column

fdr compute an FDR over selected columns. Replaces the columns with the qvalues.

Usage

Example:

```
python table2table.py --help
```

Type:

```
python table2table.py --help
```

for command line help.

Command line options

```
usage: table2table [-h] [--version]
                   [-m {transpose,normalize-by-max,normalize-by-value,multiply-by-
↪value,percentile,remove-header,normalize-by-table,upper-bound,lower-bound,kullback-
↪leibler,expand,compress,fdr,grep,randomize-rows}]
                   [-s SCALE] [-f FORMAT] [-p PARAMETERS] [-t] [--transpose]
                   [--set-transpose-field SET_TRANSPOSE_FIELD]
                   [--transpose-format {default,separated}] [--expand]
                   [--no-headers] [--columns COLUMNS] [--file FILE] [-d DELIM]
                   [-V] [--sort-by-rows SORT_ROWS] [-a VALUE]
                   [--group GROUP_COLUMN]
                   [--group-function {min,max,sum,mean,stats,cat,uniq}]
                   [--join-table JOIN_COLUMN]
                   [--collapse-table COLLAPSE_TABLE]
                   [--join-column-name JOIN_COLUMN_NAME] [--flatten-table]
                   [--as-column] [--split-fields] [--separator SEPARATOR]
                   [--fdr-method {BH,bonferroni,holm,hommel,hochberg,BY}]
                   [--fdr-add-column FDR_ADD_COLUMN] [--id-column ID_COLUMN]
                   [--variable-name VARIABLE_NAME] [--value-name VALUE_NAME]
                   [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                   [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                   [--log-config-filename LOG_CONFIG_FILENAME]
                   [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                   [-E STDERR] [-S STDOUT]
table2table: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'join_tables'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
```

(continues on next page)

(continued from previous page)

```
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'substitute_tokens'
```

Stats

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'compare_histograms'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'data2roc'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'data2stats'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
```

(continues on next page)

(continued from previous page)

```
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'data2bins'
```

data2histogram.py - histogram data in a table

Tags Python

Purpose

This script computes histograms over one or more columns of a table.

Usage

Example:

```
python data2histogram.py --help
```

Type:

```
python data2histogram.py --help
```

for command line help.

Command line options

```
usage: data2histogram [-h] [-r RANGE] [-b BIN_SIZE] [-i] [--no-null]
                      [--no-titles] [-c COLUMNS] [--min-data MIN_DATA]
                      [--min-value MIN_VALUE] [--max-value MAX_VALUE]
                      [--no-empty-bins] [--with-empty-bins] [--normalize]
                      [--cumulative] [--reverse-cumulative]
                      [--header-names HEADERS] [--ignore-out-of-range]
                      [--missing-value MISSING_VALUE] [--use-dynamic-bins]
                      [--on-the-fly] [--timeit TIMEIT_FILE]
                      [--timeit-name TIMEIT_NAME] [--timeit-header]
                      [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                      [--log-config-filename LOG_CONFIG_FILENAME]
                      [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                      [-E STDERR] [-S STDOUT]
data2histogram: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
```

(continues on next page)

(continued from previous page)

```
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'data2multiple_anova'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'histogram2histogram'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'merge_tables'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'modify_table'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'r_compare_distributions'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
```

(continues on next page)

(continued from previous page)

```
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'r_mann_whitney_u'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'r_table2scatter'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'r_test'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'calculate_histogram_2D'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'simulate_function'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'histograms2kl'
```

Tools

Cluster and jobs

split_file.py - split a file into parts

Tags Python

Purpose

Todo: describe purpose of the script.

Usage

Example:

```
python split_file.py --help
```

Type:

```
python split_file.py --help
```

for command line help.

Command line options

```
python /home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat < stdin > stdout

split a file into chunks.

OPTIONS:
-h, --help                      print this message.
-v, --verbose=                  loglevel.
-r, --split-regex               split at regular expression
-a, --after                     split after match
```

(continues on next page)

(continued from previous page)

```
-s, --skip                        do not echo match
-p, --pattern-output              pattern of output files (has to contain s)
-c, --column=                     split according to column
-m, --map=                        split according to map
-d, --dry-run                     echo files that would be created,
                                  but do not create any.
-e, --header-names                      add header to each file
-r, --remove-key                  remove key column
-append                           append data to existing files.
--pattern-identifier              if given, use this pattern to extract
                                  id from column.
--chunk-size                      Number of matching records in each output file
--version                         output version information
 option -? not recognized
```

Other

cgat_script_template.py - template for cgat scripts

Author

Tags Python

Purpose

Usage

Example:

```
python cgat_script_template.py
```

Type:

```
python cgat_script_template.py --help
```

for command line help.

Command line options

```
usage: cgat-script-template [-h] [-t TEST] [--timeit TIMEIT_FILE]
                            [--timeit-name TIMEIT_NAME] [--timeit-header]
                            [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                            [--log-config-filename LOG_CONFIG_FILENAME]
                            [--tracing {function}] [-? ?] [-I STDIN]
                            [-L STDLOG] [-E STDERR] [-S STDOUT]
cgat-script-template: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
```

(continues on next page)

(continued from previous page)

```
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'convert_time2seconds'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'set_diff'
```

Unsorted

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'add_random_reads_to_bam'
```

bam2UniquePairs.py - filter/report uniquely mapped read pairs from a (bwa!) bam-file

Tags Genomics NGS

Purpose

Utility script to report and/or filter out “uniquely mapped” properly paired reads

Reports:

1. The percentage of properly mapped read pairs with at least one uniquely mapped (XT=U) read
2. The percentage of properly mapped read pairs with at least one best mapped (X0-1) read
3. The percentage of properly mapped read pairs with at least one uniquely or best mapped (X0-1) read

If outfile is specified, reads are emitted when they are properly paired and the pair has at least one read that is either best or uniquely mapped.

Duplication is ignored.

Only BWA is supported.

TODO: cache and emit reads rather than iterating over the samfile twice...

```
usage: bam2UniquePairs [-h] [--version] [-f FILENAME] [-a ALIGNER] [-r REPORT]
                       [-o OUTFILE] [--timeit TIMEIT_FILE]
                       [--timeit-name TIMEIT_NAME] [--timeit-header]
                       [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                       [--log-config-filename LOG_CONFIG_FILENAME]
                       [--tracing {function}] [-? ?]
                       [-P OUTPUT_FILENAME_PATTERN] [-F] [-I STDIN]
                       [-L STDLOG] [-E STDERR] [-S STDOUT]
bam2UniquePairs: error: argument -?: expected one argument
```

bam2bam.py - modify bam files

Purpose

This script reads a *bam* formatted file from stdin, performs an action (see methods below) then outputs a modified *bam* formatted file on stdout.

Note: You need to redirect logging information to a file (via -L) or turn it off via -v 0 in order to get a valid sam/bam file.

Documentation

The script implements the following methods:

`set-nh`

set the NH flag. Some tools (bowtie, bwa) do not set the NH flag. If set, this option will set the NH flag (for mapped reads). This option requires the bam/sam file to be sorted by read name.

`unset-unmapped_mapq`

some tools set the mapping quality of unmapped reads. This causes a violation in the Picard tools.

`filter`

remove alignments based on a variety of flags. The filtering method is determined by the `--filter-method` option. These may be `unique`, `non-unique`, `mapped`, `NM` or `CM`. If `unique` is set, only uniquely mapping reads will be output. If `non-unique` is set then only multi-mapping reads will be output. This method first checks for the NH flag - if set, a unique match should have at most NH=1 hits. If not set, the method checks for BWA flags. Currently it checks if X0 is set (X0=Number of best hits found by BWA). If `mapped` is given, unmapped reads will be removed. If `NM` or `CM` is set, the alignment of reads in two sam files (input and reference) is compared and only reads with a lower number of mismatches in the input compared to the reference sam file will be kept. If `CM` is set, the colourspace mismatch tag (for ABI Solid reads) will be used to count differences to the reference sam file. By default, the `NM` (number of mismatches) tag is used. The tag that is used needs to present in both input sam file and the reference sam file. If `unique` is given this wil NOT remove any unmapped reads. This can be achieved by providing the `filter` option twice, once each with `mapped` and `unique`.

Note: The filter methods can't currently combined with any of the other methods - this is work in progress.

`strip-sequence`

remove the sequence from all reads in a bam-file. Note that stripping the sequence will also remove the quality scores. Stripping is not reversible if the read names are not unique.

`strip-quality`

remove the quality scores from all reads in a bam-file. Stripping is not reversible if the read names are not unique.

`set-sequence`

set the sequence and quality scores in the bam file to some dummy values ('A' for sequence, 'F' for quality which is a valid score in most fastq encodings. Necessary for some tools that can not work with bam-files without sequence.

`unstrip`

add sequence and quality scores back to a bam file. Requires a *fastq* formatted file with the sequences and quality scores to insert.

`unset-unmapped-mapq`

sets the mapping quality of unmapped reads to 0.

`keep-first-base`

keep only the first base of reads so that read counting tools will only consider the first base in the counts

`downsample-single`

generates a downsampled *bam* file by randomly subsampling reads from a single ended *bam* file. The downsmpling retains multimapping reads. The use of this requires downsampling parameter to be set and optionally randomseed.

`downsample-paired`

generates a downsampled *bam* file by randomly subsampling reads from a paired ended *bam* file. The downsampling retains multimapping reads. The use of this requires downsampling parameter to be set and optionally randomseed.

`add-sequence-error`

add a certain amount of random error to read sequences. This method picks a certain proportion of positions within a read's sequence and alters the nucleotide to a randomly chosen alternative. The model is naive and applies uniform probabilities for positions and nucleotides. The method does not update base qualities, the alignment and the NM flag. As a result, error rates that are computed via the NM flag will be unaffected. The error rate is set by –error-rate.

By default, the script works from stdin and outputs to stdout.

Usage

For example:

```
cgat bam2bam --method=filter --filter-method=mapped < in.bam > out.bam
```

will remove all unmapped reads from the bam-file.

Example for running downsample:

cgat bam2bam –method=downsample-paired –downsample=30000 –randomseed=1 -L out.log < Paired.bam > out.bam

Type:

```
cgat bam2bam --help
```

for command line help.

Command line options

```
usage: bam2bam [-h] [--version]
               [-m {filter,keep-first-base,set-nh,set-sequence,strip-sequence,strip-
↪quality,unstrip,unset-unmapped-mapq,downsample-single,downsample-paired,add-
↪sequence-error}]
               [--strip-method {all,match}]
               [--filter-method {NM,CM,mapped,unique,non-unique,remove-list,keep-list,
↪error-rate,min-read-length,min-average-base-quality}]
               [--reference-bam-file REFERENCE_BAM] [--force-output]
               [--output-sam] [--first-fastq-file FASTQ_PAIR1]
               [--second-fastq-file FASTQ_PAIR2] [--downsample DOWNSAMPLE]
               [--filename-read-list FILENAME_READ_LIST]
               [--error-rate ERROR_RATE]
               [--minimum-read-length MINIMUM_READ_LENGTH]
               [--minimum-average-base-quality MINIMUM_AVERAGE_BASE_QUALITY]
               [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
               [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
               [--log-config-filename LOG_CONFIG_FILENAME]
               [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
               [-E STDERR] [-S STDOUT]
bam2bam: error: argument -?: expected one argument
```

bam2bed.py - convert bam formatted file to bed formatted file

Tags Genomics NGS Intervals BAM BED Conversion

Purpose

This tool converts BAM files into BED files supplying the intervals for each read in the BAM file. BAM files must have a corresponding index file ie. example.bam and example.bam.bai

For example:

```
samtools view example.bam

READ1    163    1      13040   15     76M    =      13183   219     ...
READ1    83     1      13183   7      76M    =      13040   -219    ...
READ2    147    1      13207   0      76M    =      13120   -163    ...

python bam2bed.py example.bam

1       13039   13115   READ1     15      +
1       13119   13195   READ2     0       +
1       13182   13258   READ1     7       -
1       13206   13282   READ2     0       -
```

By default, bam2bed outputs each read as a separate interval. With the option `--merge-pairs` paired-end reads are merged and output as a single interval. The strand is set according to the first read in a pair.

Usage

```
cgat bam2bed BAMFILE [--merge-pairs] [options]
```

operates on the file BAMFILE:

```
cgat bam2bed [--merge-pairs] [options]
```

operates on the stdin as does:

```
cgat bam2bed -I BAMFILE [--merge-pairs] [options]
```

To merge paired-end reads and output fragment interval ie. leftmost mapped base to rightmost mapped base:

```
cat example.bam | cgat bam2bed --merge-pairs

1       13119   13282   READ2     0       +
1       13039   13258   READ1     7       +
```

To use merge pairs on only a region of the genome use samtools view:

```
samtools view -ub example.bam 1:13000:13100 | cgat bam2bed --merge-pairs
```

Note that this will select fragments were the first read-in-pair is in the region.

Options

-m, --merge-pairs Output one region per fragment rather than one region per read, thus a single region is create stretching from the start of the frist read in pair to the end of the second.

Read pairs that meet the following criteria are removed:

- Reads where one of the pair is unmapped
- Reads that are not paired
- Reads where the pairs are mapped to different chromosomes
- Reads where the the insert size is not between the max and min (see below)

Warning: Merged fragements are always returned on the +ve strand. Fragement end point is estimated as the alignment start position of the second-in-pair read + the length of the first-in-pair read. This may lead to inaccuracy if you have an intron-aware aligner.

--max-insert-size, --min-insert-size The maximum and minimum size of the insert that is allowed when using the –merge-pairs option. Read pairs closer to gether or futher apart than the min and max repsectively are skipped.

-b, --bed-format What format to output the results in. The first n columns of the bed file will be output.

Type:

```
python bam2bed.py --help
```

for command line help.

Command line options

```
usage: bam2bed [-h] [--version] [-m] [--max-insert-size MAX_INSERT_SIZE]
               [--min-insert-size MIN_INSERT_SIZE] [--bed-format {3,4,5,6}]
               [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
               [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
               [--log-config-filename LOG_CONFIG_FILENAME]
               [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
               [-E STDERR] [-S STDOUT]
bam2bed: error: argument -?: expected one argument
```

bam2fastq.py - output fastq files from a bam-file

Tags Genomics NGS Sequences BAM FASTQ Conversion

Purpose

This script takes a *bam* formatted file and converts to it to one or two *fastq* formatted files for single-end or paired-end data, respectively.

For paired-end data, the first fastq file contains the first read of a read pair and the other contains the second read of read pair.

Example

For example:

```
cat in.bam cgat bam2fastq out.1.fastq.gz out.2.fastq.gz
```

This command converts the *bam* formatted file in.bam into *fastq* files containing forward reads (out.1.fastq.gz) and reverse reads (out.2.fastq.gz). The output files can alternatively supplied via the option `--output-pattern-filename`. The statement below will create the same two output files:

```
cat in.bam cgat bam2fastq --output-filename-pattern=out.%s.fastq.gz
```

Type:

```
python bam2fastq.py --help
```

for command line help.

Command line options

```
usage: bam2fastq [-h] [--version] [--timeit TIMEIT_FILE]
                 [--timeit-name TIMEIT_NAME] [--timeit-header]
                 [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                 [--log-config-filename LOG_CONFIG_FILENAME]
                 [--tracing {function}] [-? ?] [-P OUTPUT_FILENAME_PATTERN]
                 [-F] [-I STDIN] [-L STDLOG] [-E STDERR] [-S STDOUT]
bam2fastq: error: argument -?: expected one argument
```

bam2peakshape.py - compute peak shape features from a bam-file

Tags Genomics NGS Intervals BAM BED Summary

Purpose

This script takes a *bed* formatted file with regions of interest, for example binding intervals from a ChIP-Seq experiment. Using a collection of aligned reads is a *bam* formatted file or *bigwig* formatted file, the script outputs a collection of features describing the peak shape.

This script is designed with a slight emphasis on ChIP-Seq datasets. The main reason that this script is better suited for ChIP-Seq is that(1) it is able to center the counting window at the summit of every individual peak; (2) it is also able to use the control ChIP-Seq library to enable side-by-side comparison of treatment vs control;(3) it can randomly shift the set of input regions to generate a artificial set of regions, in the absence of real ChIP-Seq control library, the random regions can provide a peaks profile that can be used as the control.

For example, given the peaks regions defined by analyzing some ChIP-Seq dataset (e.g. by using MACS), and without the need to use any additional genomic annotations (e.g. ENSEMBL, refseq), we can visualise the binding profiles of transcriptionfactors ChIP-Seq data relative to the center of each peak regions.

The script outputs a tab-separated table on stdout containing features for each interval. A peak is defined as the location of the highest density in an interval. The width of the peak (peak_width) is defined as the region around the peak in which the density does not drop below a threshold of peak_heigt * 90%.

Usage

Detailed usage example

The following command will generate the peak shape plot for the peak regions defined in `onepeak.bed`, using the reads stored in `small.bam`. The command will also create a profile for the control library. The control library in this example is re-using the same reads file `small.bam`, however, in your actual experiment, it should be a different library (the input library for this ChIP-Seq experiment).:

```
python ./scripts/bam2peakshape.py ./tests/bam2peakshape.py/small.bam .
↪/tests/bam2peakshape.py/onepeak.bed --control-bam-file=./tests/
↪bam2peakshape.py/small.bam --use-interval --normalize-transcript
```

Output files

Among the features output are:

Column	*Content*
peak_height	number of reads at peak
peak_median	median coverage compared to peak height
interval_width	width of interval
peak_width	width of peak
bins	bins for a histogram of densities within the interval.
npeaks	number of density peaks in interval.
peak_center	point of highest density in interval
peak_relative_pos	point of highest density in interval coordinates
counts	counts for a histogram of densities within the interval
furthest_half_heigh	Distance of peak center to furthest half-height position
closest_half_height	Distance of peak center to closest half-height position

Additionally, the script outputs a set of matrixes with densities over intervals that can be used for plotting. The default filenames are `(matrix|control)_<sortorder>.tsv.gz`, The names can be controlled with the `--output-filename-pattern` option.

Type:

```
python bam2peakshape.py --help
```

for command line help.

Options

Option: Shift

shift the each read by a certain distance, because in a ChIP-Seq experment, the read is always at the edge of an sonicated fragment, the actual binding site is usually L/2 distance away from the read, where L is the length of sonicated fragment (determined either experimentally or computationally).

This option is used only if the input reads are in *bam* formatted file. If input reads are *bigwig* formatted file, this option is ignored.

Option: Random shift

randomly shift the set of input regions to generate a artificial set of regions. In the absence of real ChIP-Seq control library, the random regions can provide a peaks profile that can be used as the control.

Option: Centring method

"reads" will output in the way that the summit of the peaks are aligned. "middle" will output in the way that the middle of the input bed intervals are aligned.

Option: Only interval

Only count reads that are in the interval as defined by the input bed file.

Option: normalization=sum

normalize counts such that the sum of all counts in all features are exactly 1000000.

The detail normalization algorithm as follows: norm = sum(all counts in all features)/1000000.0 normalized count = normalized count / norm

Todo: paired-endedness is not fully implemented.

Command line options

```
usage: bam2peakshape [-h] [--version] [-f {bam,bigwig}] [-o] [-w WINDOW_SIZE]
                     [-b BIN_SIZE] [--smooth-method {none,sum,sg}]
                     [-s {peak-height,peak-width,unsorted,interval-width,interval-
↪score}]
                     [-c CONTROL_FILES] [-r] [-e {reads,middle}]
                     [-n {none,sum}] [--use-strand] [-i SHIFT]
                     [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                     [--timeit-header] [--random-seed RANDOM_SEED]
                     [-v LOGLEVEL] [--log-config-filename LOG_CONFIG_FILENAME]
                     [--tracing {function}] [-? ?]
                     [-P OUTPUT_FILENAME_PATTERN] [-F] [-I STDIN] [-L STDLOG]
```

(continues on next page)

(continued from previous page)

```
                      [-E STDERR] [-S STDOUT]
bam2peakshape: error: argument -?: expected one argument
```

compute stats from a bam-file

Purpose

This script takes a bam file as input and computes a few metrics by iterating over the file. The metrics output are:

Category	*Content*
total	total number of alignments in bam file
alignments_mapped	alignments mapped to a chromosome (bam flag)
alignments_unmapped	alignments unmapped (bam flag)
qc_fail	alignments failing QC (bam flag)
mate_unmapped	alignments in which the mate is unmapped (bam flag)
reverse	alignments in which read maps to reverse strand (bam flag)
mate_reverse	alignments in which mate maps to reverse strand (bam flag)
proper_pair	alignments in which both pairs have been mapped properly (according to the mapper) (bam flag)
read1	alignments for 1st read of pair (bam flag)
paired	alignments of reads that are paired (bam flag)
duplicate	read is PCR or optical duplicate (bam flag)
read2	alignment is for 2nd read of pair (bam flag)
secondary	alignment is not primary alignment
alignments_duplicates	number of alignments mapping to the same location
alignments_unique	number of alignments mapping to unique locations
reads_total	number of reads in file. Either given via –num-reads or deduc ed as the sum of mappend and unmapped reads
reads_mapped	number of reads mapping in file. Derived from the total number o f alignments and removing counts for multiple matches. Requires the NH flag to be set correctly.
reads_unmapped	number of reads unmapped in file. Assumes that there is only one entry per unmapped read.
reads_missing	number of reads missing, if number of reads given by –input-rea ds. Otherwise 0.
pairs_total	number of total pairs - this is the number of reads_total divided by two. If there were no pairs, pairs_total will be 0.
pairs_mapped	number of mapped pairs - this is the same as the number of proper pairs.

Additionally, the script outputs histograms for the following tags and scores.

- NM: number of mismatches in alignments.
- NH: number of hits of reads.
- mapq: mapping quality of alignments.

Supplying a fastq file

If a fastq file is supplied (`--fastq-file`), the script will compute some additional summary statistics. However, as it builds a dictionary of all sequences, it will also require a good amount of memory. The additional metrics output are:

Category	*Content*
pairs_total	total number of pairs in input data
pairs_mapped	pairs in which both reads map
pairs_unmapped	pairs in which neither read maps
pairs_proper_unique	pairs which are proper and map uniquely.
pairs_incomplete_unique	pairs in which one of the reads maps uniquely, but the other does not map.
pairs_incomplete_multimapping	pairs in which one of the reads maps uniquely, but the other maps to multiple locations.
pairs_proper_duplicate	pairs which are proper and unique, but marked as duplicates.
pairs_proper_multimapping	pairs which are proper, but map to multiple locations.
pairs_not_proper_unique	pairs mapping uniquely, but not flagged as proper
pairs_other	pairs not in any of the above categories

Note that for paired-end data, any or `/2` suffixes will be removed from the read name in the assumption that these have been removed in the bam file as well.

Usage

Example:

```
python bam2stats.py in.bam
```

This command will generate various statistics based on the supplied BAM file, such as percentage reads mapped and percentage reads mapped in pairs. The output looks like this:

category	counts	percent	of
alignments_total	32018	100.00	alignments_total
alignments_mapped	32018	100.00	alignments_total
alignments_unmapped	0	0.00	alignments_total
alignments_qc_fail	0	0.00	alignments_mapped
alignments_mate_unmapped	241	0.75	alignments_mapped
alignments_reverse	16016	50.02	alignments_mapped
alignments_mate_reverse	15893	49.64	alignments_mapped
alignments_proper_pair	30865	96.40	alignments_mapped
alignments_read1	16057	50.15	alignments_mapped
alignments_paired	32018	100.00	alignments_mapped
alignments_duplicate	0	0.00	alignments_mapped
alignments_read2	15961	49.85	alignments_mapped
alignments_secondary	0	0.00	alignments_mapped
alignments_filtered	31950	99.79	alignments_mapped
reads_total	34250	100.00	reads_total
reads_unmapped	0	0.00	reads_total
reads_mapped	32018	93.48	reads_total
reads_missing	2232	6.52	reads_total

continues on next page

Table 1 – continued from previous page

reads_mapped_unique	32018	100.00	reads_mapped
reads_multimapping	0	0.00	reads_mapped
pairs_total	17125	100.00	pairs_total
pairs_mapped	17125	100.00	pairs_total
pairs_unmapped	0	0.00	pairs_total
pairs_proper_unique	14880	86.89	pairs_total
pairs_incomplete_unique	2232	13.03	pairs_total
pairs_incomplete_multimapping	0	0.00	pairs_total
pairs_proper_duplicate	0	0.00	pairs_total
pairs_proper_multimapping	0	0.00	pairs_total
pairs_not_proper_unique	13	0.08	pairs_total
pairs_other	0	0.00	pairs_total
read1_total	17125	100.00	read1_total
read1_unmapped	0	0.00	read1_total
read1_mapped	16057	93.76	read1_total
read1_mapped_unique	16057	100.00	read1_mapped
reads_multimapping	0	0.00	read1_mapped
read1_missing	1068	6.65	read1_total
read2_total	17125	100.00	read2_total
read2_unmapped	0	0.00	read2_total
read2_mapped	15961	93.20	read2_total
read2_mapped_unique	15961	100.00	read2_mapped
reads_multimapping	0	0.00	read2_mapped
read2_missing	1164	7.29	read2_total

The first column contains the caterogy, the second the number of counts and the third a percentage. The fourth column denotes the denominator that was used to compute the percentage. In the table above, wee see that 16,057 first reads in a pair map and 15,961 second reads in pair map, resulting in 14,880 proper uniquely mapped pairs.

Type:

```
cgat bam2stats --help
```

for command line help.

Bam2stats can read from standard input:

```
cat in.bam | python bam2stats.py -
```

Documentation

Reads are not counted via read name, but making use of NH and HI flags when present. To recap, NH is the number of reported alignments that contain the query in the current record, while HI is the hit index and ranges from 0 to NH-1.

Unfortunately, not all aligners follow this convention. For example, gsnap seems to set NH to the number of reportable alignments, while the actual number of reported alignments in the file is less. Thus, if the HI flag is present, the maximum HI is used to correct the NH flag. The assumption is, that the same reporting threshold has been used for all alignments.

If no NH flag is present, it is assumed that all reads have only been reported once.

Multi-matching counts after filtering are really guesswork. Basically, the assumption is that filtering is consistent and will tend to remove all alignments of a query.

The error rates are computed using the following key:

substitution_rate Number of mismatches divided by number of aligned bases. This is the same as the samtools stats error rate.

insertion_rate Number of deletions in the read/insertions in the reference divided by the number of aligned bases.

deletion_rate Number of insertions in the read/deletions in the reference divided by the number of aligned bases.

error_rate Number of mismatches and deletions in the read divided by the number of aligned bases.

coverage Percentage of bases aligned divided by read length.

The following graphic illustrates the computation. A . signifies a position that is included in the metric with *X* being an error:

```
AAAAACAAAA AAAAAAAA   Reference
 AAAAAAAAAAAA AAA     Read
 ....X.... ......     substitution_rate NM / (CMATCH + CINS)
 .........X.. ...     insertion_rate CINS / (CMATCH + CINS)
 ......... ..X...     deletion_rate CDEL / (CMATCH + CDEL)
 ....X.... ..X...     error_rate NM / (CMATCH + CINS) (corresponds to samtools stats)
 .........X.. ...     match_rate CMATCH / (CMATCH + CINS)
 ....X.... .. ...     mismatch_rate NM / (CMATCH) (1 - percent_identity/100)
```

With CINS: Insertion into the reference (consumes read, but not reference) and CDEL=Deletion from the reference (consumes reference, but not read).

Command line options

```
usage: bam2stats [-h] [--version] [-r GFF] [-f] [-i INPUT_READS] [-d]
                 [--output-readmap] [--add-alignment-details]
                 [-q FILENAME_FASTQ] [--basic-counts] [--timeit TIMEIT_FILE]
                 [--timeit-name TIMEIT_NAME] [--timeit-header]
                 [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                 [--log-config-filename LOG_CONFIG_FILENAME]
                 [--tracing {function}] [-? ?] [-P OUTPUT_FILENAME_PATTERN]
                 [-F] [-I STDIN] [-L STDLOG] [-E STDERR] [-S STDOUT]
bam2stats: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'bam2transcriptContribution'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
```

(continues on next page)

(continued from previous page)

```
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'barplotGo'
```

beds2counts - compute overlap stats between multiple bed files

Tags Genomics Intervals Comparison BED Counting

Purpose

This script takes multiple bed files e.g. from multiple samples from the same experiment. It assesses the overlap between samples and outputs a count for each merged interval corresponding to the number of samples that a particular interval was found in.

Example

For example if the command:

```
cgat bed2counts a.bed b.bed c.bed > output.tsv
```

is run, where a.bed-c.bed look like:

```
                1         2         3         4
       01234567890123456789012345678901234567890123
a.bed: -------           -----               -------
b.bed:      -----         --
c.bed:   ---

Union: ----------        -----               -------
```

Then output.tsv will look like:

```
contig      start   end     count
chr1        0       7       3
chr1        17      22      2
chr1        37      44      1
```

Options

The only option other than the standard cgat options is -i, –bed-file this allows the input files to be provided as a comma seperated list to the option rather than a space delimited set of positional arguements. It is present purely for galaxy compatibility.

Usage

cgat beds2counts BED [BED . . .] [OPTIONS]

Command line options

```
usage: beds2counts [-h] [--version] [--bed-file bed] [--timeit TIMEIT_FILE]
                   [--timeit-name TIMEIT_NAME] [--timeit-header]
                   [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                   [--log-config-filename LOG_CONFIG_FILENAME]
                   [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                   [-E STDERR] [-S STDOUT]

beds2counts - compute overlap stats between multiple bed files
================================================================

:Tags: Genomics Intervals Comparison BED Counting

Purpose
-------

This script takes multiple bed files e.g. from multiple samples from
the same experiment. It assesses the overlap between samples and
outputs a count for each merged interval corresponding to the number
of samples that a particular interval was found in.

Example
-------

For example if the command::

    cgat bed2counts a.bed b.bed c.bed > output.tsv

is run, where a.bed-c.bed look like::

                     1         2         3         4
           01234567890123456789012345678901234567890123
    a.bed: -------          -----               -------
    b.bed:      -----        --
    c.bed:  ---

    Union: ----------       -----               -------

Then output.tsv will look like::

    contig        start        end        count
    chr1        0        7        3
    chr1        17        22        2
    chr1        37        44        1

Options
-------

The only option other than the standard cgat options is -i, --bed-file this
allows the input files to be provided as a comma seperated list to the option
rather than a space delimited set of positional arguements. It is present
```

(continues on next page)

(continued from previous page)

```
purely for galaxy compatibility.

Usage
-----

    cgat beds2counts BED [BED ...] [OPTIONS]

Command line options
--------------------

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  --bed-file bed        supply list of bed files (default: [])

Script timing options:
  --timeit TIMEIT_FILE  store timeing information in file. (default: None)
  --timeit-name TIMEIT_NAME
                        name in timing file for this class of jobs (default:
                        all)
  --timeit-header       add header for timing information. (default: None)

Common options:
  --random-seed RANDOM_SEED
                        random seed to initialize number generator with
                        (default: None)
  -v LOGLEVEL, --verbose LOGLEVEL
                        loglevel. The higher, the more output. (default: 1)
  --log-config-filename LOG_CONFIG_FILENAME
                        Configuration file for logger. (default: logging.yml)
  --tracing {function}  enable function tracing. (default: None)
  -? ?                  output short help (command line options only.
                        (default: None)

Input/output options:
  -I STDIN, --stdin STDIN
                        file to read stdin from. (default: <_io.TextIOWrapper
                        name='<stdin>' mode='r' encoding='UTF-8'>)
  -L STDLOG, --log STDLOG
                        file with logging information. (default:
                        <_io.TextIOWrapper name='<stdout>' mode='w'
                        encoding='UTF-8'>)
  -E STDERR, --error STDERR
                        file with error information. (default:
                        <_io.TextIOWrapper name='<stderr>' mode='w'
                        encoding='UTF-8'>)
  -S STDOUT, --stdout STDOUT
                        file where output is to go. (default:
                        <_io.TextIOWrapper name='<stdout>' mode='w'
                        encoding='UTF-8'>)
```

bed2fasta.py - get sequences from bed file

Tags Genomics Intervals Sequences Conversion BED FASTA

Purpose

This script outputs nucleotide sequences for intervals within a *bed* formatted file using a corresponding genome file.

Usage

A required input to bed2fasta.py is a cgat indexed genome. To obtain an idexed human reference genome we would type

Example:: cat hg19.fasta | index_fasta.py hg19 > hg19.log

This file would then serve as the –genome-file when we wish to extract sequences from a *bed* formatted file.

For example we could now type:

```
cat in.bed | python bed2fasta.py --genome-file hg19 > out.fasta
```

Where we take a set of genomic intervals (e.g. from a human ChIP-seq experiment) and output their respective nucleotide sequences.

Type:

```
python bed2fasta.py --help
```

for command line help.

Command line options

```
usage: bed2fasta [-h] [-g GENOME_FILE] [-m {dust,dustmasker,softmask,none}]
                 [--output-mode {intervals,leftright,segments}]
                 [--min-sequence-length MIN_LENGTH]
                 [--max-sequence-length MAX_LENGTH]
                 [--extend-at {none,3,5,both,3only,5only}]
                 [--extend-by EXTEND_BY] [--use-strand] [--timeit TIMEIT_FILE]
                 [--timeit-name TIMEIT_NAME] [--timeit-header]
                 [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                 [--log-config-filename LOG_CONFIG_FILENAME]
                 [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                 [-E STDERR] [-S STDOUT]
bed2fasta: error: argument -?: expected one argument
```

bed2stats.py - summary of bed file contents

Tags Genomics Intervals Summary BED

Purpose

This script takes a *bed*-formatted file as input and outputs the number of intervals and bases in the bed file. Counts can be subdivided by setting the `--aggregate-by` command line option:

`contig` output counts per contig (column 1)

`name` output counts grouped by the name field in the *bed* formatted file (column 4)

`track` output counts per track in the *bed* formatted file.

Note that a count of bases usually makes only sense if the intervals submitted are non-overlapping.

If the option –add-percent is given, an additional column will output the percent of the genome covered by intervals. This requires a –genome-file to be given as well.

Usage

To count the number of intervals, type:

```
cgat bed2table < in.bed
```

track	ncontigs	nintervals	nbases
all	23	556	27800

To count per contig:

```
cgat bed2table --aggregate=contig < in.bed
```

track	ncontigs	nintervals	nbases
chrX	1	11	550
chr13	1	12	600
chr12	1	37	1850
…	…	…	…

Type:

```
cgat bed2table --help
```

for command line help.

Command line options

```
usage: bed2stats [-h] [-g GENOME_FILE] [-a {name,contig,track,none}] [-p]
                 [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                 [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                 [--log-config-filename LOG_CONFIG_FILENAME]
                 [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                 [-E STDERR] [-S STDOUT]
bed2stats: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 129, in main
    module = imp.load_module(command, file, pathname, description)
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 235, in load_module
    return load_source(name, filename, file)
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 172, in load_source
    module = _load(spec)
  File "<frozen importlib._bootstrap>", line 696, in _load
  File "<frozen importlib._bootstrap>", line 677, in _load_unlocked
  File "<frozen importlib._bootstrap_external>", line 728, in exec_module
  File "<frozen importlib._bootstrap>", line 219, in _call_with_frames_removed
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/tools/bed2table.py", line 100, in <module>
    import cgat.SequenceProperties as SequenceProperties
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/SequenceProperties.py", line 49, in <module>
    import Bio.Alphabet.IUPAC
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/Bio/Alphabet/__init__.py", line 21, in <module>
    "Bio.Alphabet has been removed from Biopython. In many cases, the alphabet can␣
↪simply be ignored and removed from scripts. In a few cases, you may need to specify␣
↪the ``molecule_type`` as an annotation on a SeqRecord for your script to work␣
↪correctly. Please see https://biopython.org/wiki/Alphabet for more information."
ImportError: Bio.Alphabet has been removed from Biopython. In many cases, the␣
↪alphabet can simply be ignored and removed from scripts. In a few cases, you may␣
↪need to specify the ``molecule_type`` as an annotation on a SeqRecord for your␣
↪script to work correctly. Please see https://biopython.org/wiki/Alphabet for more␣
↪information.
```

beds2beds.py - decompose bed files

Tags Genomics Intervals BED Manipulation

Purpose

This script will decompose a collection of input bedfiles into a collection of unions or intersections.

Options

Files are collected by a regular expression pattern given to the option `--pattern-identifier`.

The script behaviour is determined by the `--method` option with either of the following choices:

`merged-combinations` merge intervals across *bed* files and only report those that appear in every file.

`unmerged-combinations` for each *bed* file, report intervals that overlap with intervals in every other *bed* file.

If the `--exclusive-overlap` option is set, report exclusive overlap. Only intervals will be reported that overlap in a pairwise comparison but do not overlap with intervals in any of the other sets.

This script requires bed files indexed by tabix.

Usage

For example, you have ChIP-Seq data for PolII and two transcription factors tf1 and tf2. The following statement will output four *bed* files:

```
zcat polii.bed.gz | head

chr17    1    100    8    1
chr19   -50    50    6    1
chr19    0    100    1    1
chr19    50   150    1    1
chr19   150   200    2    1
chr19   201   300    3    1

python beds2beds.py polii.bed.gz tf1.bed.gz tf2.bed.gz

zcat tf1.bed.gz | head

chr1    35736     40736    ENST000004173240    -
chr1    60881     65881    ENST000005349900    +
chr1    64090     69090    ENST000003351370    +
chr1    362658    367658   ENST000004264060    +
chr1    622034    627034   ENST000003328310    -
chr1    716405    721405   ENST000003585330    +
```

The four files contain intervals, that

1. have PolII and tf1 present,
2. have PolII and tf2 present,
3. have tf1 and tf2 present, or
4. have PolII and tf1 and tf2 present.

If the –exclusive-overlap option is set, three sets will be output with intervals that

1. have PolII and tf1 present but no tf2,
2. have PolII and tf2 present but no tf1,
3. have tf1 and tf2 present but no PolII.

Type:

```
python beds2beds.py --help
```

for command line help.

Command line options

```
usage: beds2beds [-h] [--version] [-e] [-p PATTERN_ID]
                 [-m {merged-combinations,unmerged-combinations}]
                 [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                 [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                 [--log-config-filename LOG_CONFIG_FILENAME]
                 [--tracing {function}] [-? ?] [-P OUTPUT_FILENAME_PATTERN]
                 [-F] [-I STDIN] [-L STDLOG] [-E STDERR] [-S STDOUT]
beds2beds: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'blast2table'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'chain2stats'
```

combine_tables.py - join tables

Tags Python

Purpose

This script reads several tab-separated tables and joins them into a single one.

Todo:

- Rename to tables2table.py
- Use pandas dataframes for fast IO and merging/joining

Usage

The option `--header-names` sets the column titles explicitely. Add `--skip-titles` if you want to avoid echoing the original title in the input files.

Example:

```
python combine_tables.py --help
```

Type:

```
python combine_tables.py --help
```

for command line help.

Command line options

```
usage: combine-tables [-h] [--version] [-t] [--ignore-titles] [-i]
                      [-m MISSING_VALUE] [--header-names HEADERS] [-c COLUMNS]
                      [-k TAKE] [-g GLOB] [-s SORT] [-e] [-a CAT]
                      [--sort-keys {numeric,alphabetic}] [--keep-empty]
                      [--ignore-empty] [--add-file-prefix] [--use-file-prefix]
                      [--prefixes PREFIXES] [--regex-filename REGEX_FILENAME]
                      [--regex-start REGEX_START] [--regex-end REGEX_END]
                      [--test TEST] [--timeit TIMEIT_FILE]
                      [--timeit-name TIMEIT_NAME] [--timeit-header]
                      [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                      [--log-config-filename LOG_CONFIG_FILENAME]
                      [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                      [-E STDERR] [-S STDOUT]
combine-tables: error: argument -?: expected one argument
```

diff_chains.py - compare to chain formatted files

Tags Genomics GenomeAlignment CHAIN Comparison

Purpose

Compare two genomic alignment files and calculate statistics from the comparison.

Documentation

Operates on two chain formatted files.

Outputs a table with the following columns:

Column	*Content*
contig1	contig name
contig2	contig name
strand	strand
mapped1	mapped residues
identical1	identically mapped residues
different1	differently mapped residues
unique1	residues mapped only from set1
pmapped1	percentage of mapped residues
pidentical1	percentage of identically mapped residues
pdifferent1	percentage of differently mapped residues

Similar columns exist for data set 2

Usage

Example:

```
cgat diff_chains.py hg19ToMm10v1.chain.over.gz hg19ToMm10v2.chain.over.gz
```

This will compare the locations that regions within the genome hg19 map to between two different mappings to the genome mm10.

Type:

```
python diff_chains.py --help
```

for command line help.

Command line options

```
usage: diff-chains [-h] [--version] [-m] [-a] [-u] [-r RESTRICT]
                   [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                   [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                   [--log-config-filename LOG_CONFIG_FILENAME]
                   [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                   [-E STDERR] [-S STDOUT]
diff-chains: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 129, in main
    module = imp.load_module(command, file, pathname, description)
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 235, in load_module
    return load_source(name, filename, file)
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 172, in load_source
    module = _load(spec)
  File "<frozen importlib._bootstrap>", line 696, in _load
  File "<frozen importlib._bootstrap>", line 677, in _load_unlocked
  File "<frozen importlib._bootstrap_external>", line 728, in exec_module
  File "<frozen importlib._bootstrap>", line 219, in _call_with_frames_removed
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/tools/fasta2table.py", line 118, in <module>
    import cgat.SequenceProperties as SequenceProperties
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/SequenceProperties.py", line 49, in <module>
    import Bio.Alphabet.IUPAC
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/Bio/Alphabet/__init__.py", line 21, in <module>
    "Bio.Alphabet has been removed from Biopython. In many cases, the alphabet can␣
↪simply be ignored and removed from scripts. In a few cases, you may need to specify␣
↪the ``molecule_type`` as an annotation on a SeqRecord for your script to work␣
↪correctly. Please see https://biopython.org/wiki/Alphabet for more information."
ImportError: Bio.Alphabet has been removed from Biopython. In many cases, the␣
↪alphabet can simply be ignored and removed from scripts. In a few cases, you may␣
↪need to specify the ``molecule_type`` as an annotation on a SeqRecord for your␣
↪script to work correctly. Please see https://biopython.org/wiki/Alphabet for more␣
↪information.
```

fasta2variants.py - create sequence variants from a set of sequences

Tags Genomics Sequences Variants Protein FASTA Transformation

Purpose

This script reads a collection of sequences in *fasta* format and outputs a table of possible variants. It outputs for each position in a protein sequence the number of variants.

If the input sequences are nucleotide coding (CDS) sequences, for each variant a weight is output indicating the number of times that variant can occur from single nucleotide changes.

Usage

Example:

```
python fasta2variants.py -I CCDS_nucleotide.current.fna.gz -L CDS.log -S CDS.output -c
```

This will take a CDS file as input, save the log and output files, and count variants based on single nucleotide changes using the -c option.

Type:

```
python fasta2variants.py --help
```

for command line help.

Compressed (.gz) and various fasta format files (.fasta, .fna) are accepted. If the -c option is specified and the file is not a CDS sequence the script will throw an error ('length of sequence '<input_file>' is not a multiple of 3').

Command line options

```
usage: fasta2variants [-h] [--version] [-c] [--timeit TIMEIT_FILE]
                      [--timeit-name TIMEIT_NAME] [--timeit-header]
                      [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                      [--log-config-filename LOG_CONFIG_FILENAME]
                      [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                      [-E STDERR] [-S STDOUT]
fasta2variants: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'fastq2N'
```

fastq2fastq.py - manipulate fastq files

Tags Genomics NGS Sequences FASTQ Manipulation

Purpose

This script performs manipulations on *fastq* formatted files. For example it can be used to change the quality score format or sample a subset of reads.

The script predominantly is used for manipulation of single fastq files. However, for some of its functionality it will take paired data using the `--pair-fastq-file` and `--output-filename-pattern` options. This applies to the `sample` and `sort` methods.

Usage

Example:: In this example we randomly sample 50% of reads from paired data provided in two *fastq* files.

head in.fastq.1

@SRR111956.1 HWUSI-EAS618:7:1:27:1582 length=36 CCCCCCCCCCCCCCC-CCCCCCCCCCCCCCCCCCCCCCCC +SRR111956.1 HWUSI-EAS618:7:1:27:1582 length=36 =@A@9@BAB@;@BABA?=;@@BB<A@9@;@2>@;?? @SRR111956.2 HWUSI-EAS618:7:1:29:1664 length=36 CCCCCCCCCCCCCCCCCCCCCCCC-CCCCACCCCCCCC +SRR111956.2 HWUSI-EAS618:7:1:29:1664 length=36 =B@9@0>A<B=B=AAA?;*(@A>(@<=*9=9@BA>7 @SRR111956.3 HWUSI-EAS618:7:1:38:878 length=36 AGTGAGCAGGGAAACAATGTCTGTCTAAGAATTTGA

head in.fastq.2

+SRR111956.3 HWUSI-EAS618:7:1:38:878 length=36 <?@BA?;A=@BA>;@@7################## @SRR111956.4 HWUSI-EAS618:7:1:38:1783 length=36 ATTAGTATTATC-CATTTATATAATCAATAAAAATGT +SRR111956.4 HWUSI-EAS618:7:1:38:1783 length=36 ?ABBA2CCBBB2?=BB@C>=AAC@A=CBB####### @SRR111956.5 HWUSI-EAS618:7:1:39:1305 length=36 CCCCCCCCCCCCCCCCCCCCCCCC-CCCCCCCCCCCCC +SRR111956.5 HWUSI-EAS618:7:1:39:1305 length=36 AA>5;A>*91?=AAA@@BBA<B=?ABA>2>?A<BB@

command-line::

cat in.fastq.1 | python fastq2fastq.py –method=sample –sample-size 0.5 –pair-fastq-file in.fastq.2 –output-filename-pattern out.fastq.2 > out.fastq.1

head out.fastq.1 @SRR111956.1 HWUSI-EAS618:7:1:27:1582 length=36 CCCCCCCCCCCCCCC-CCCCCCCCCCCCCCCCCCCCCCCC + =@A@9@BAB@;@BABA?=;@@BB<A@9@;@2>@;?? @SRR111956.2 HWUSI-EAS618:7:1:29:1664 length=36 CCCCCCCCCCCCCCCCCCCCCCCC-CCCACCCCCCCC + =B@9@0>A<B=B=AAA?;*(@A>(@<=*9=9@BA>7 @SRR111956.3 HWUSI-EAS618:7:1:38:878 length=36 AGTGAGCAGGGAAACAATGTCTGTCTAA-GAATTTGA + <?@BA?;A=@BA>;@@7################## @SRR111956.4 HWUSI-EAS618:7:1:38:1783 length=36 ATTAGTATTATCCATTTATATAATCAATAAAAATGT + ?ABBA2CCBBB2?=BB@C>=AAC@A=CBB#######

Options

The following methods are implemented (`--method`).

`change-format`

change the quality format to new format given as target-format. Options are `sanger`, `solexa`, `phred64`, `integer` and `illumina-1.8`

`sample`

Sub-sample a fastq file. The size of the sample is set by –sample-size

`unique`

Remove duplicate reads based on read name

`trim3`

Trim a fixed number of nucleotides from the 3' end of reads. (see `--num-bases`). Note that there are better tools for trimming.

`trim5`

Trim a fixed number of nucleotides from the 5' end of reads. (see `--num-bases`). Note that there are better tools for trimming.

`sort`

Sort the fastq file by read name.

`renumber-reads`

Rename the reads based on pattern given in `--pattern-identifier` e.g. `--pattern-identifier="read_%010i"`

Type:

```
python fastq2fastq.py --help
```

for command line help.

Command line options

```
usage: fastq2fastq [-h] [--version] [-i INPUT_FASTQ_FILE]
                   [--output-removed-tsv OUTPUT_REMOVED_TSV]
                   [--output-stats-tsv OUTPUT_STATS_TSV]
                   [--output-removed-fastq OUTPUT_REMOVED_FASTQ]
                   [-m {filter-N,filter-identifier,filter-ONT,offset-quality,apply,
↪change-format,renumber-reads,sample,sort,trim3,trim5,unique,reverse-complement,grep}
↪]
                   [--set-prefix SET_PREFIX]
                   [--input-filter-tsv INPUT_FILTER_TSV]
                   [--min-average-quality MIN_AVERAGE_QUALITY]
                   [--min-sequence-length MIN_SEQUENCE_LENGTH]
                   [--quality-offset QUALITY_OFFSET]
                   [--target-format {sanger,solexa,phred64,integer,illumina-1.8}]
```

(continues on next page)

(continued from previous page)

```
                    [--guess-format {sanger,solexa,phred64,integer,illumina-1.8}]
                    [--sample-size SAMPLE_SIZE] [--pair-fastq-file PAIR]
                    [--map-tsv-file MAP_TSV_FILE] [--num-bases NBASES]
                    [--seed SEED] [--pattern-identifier RENUMBER_PATTERN]
                    [--grep-pattern GREP_PATTERN] [--timeit TIMEIT_FILE]
                    [--timeit-name TIMEIT_NAME] [--timeit-header]
                    [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                    [--log-config-filename LOG_CONFIG_FILENAME]
                    [--tracing {function}] [-? ?] [-P OUTPUT_FILENAME_PATTERN]
                    [-F] [-I STDIN] [-L STDLOG] [-E STDERR] [-S STDOUT]
fastq2fastq: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'fastq2solid'
```

fastq2table.py - compute stats on reads in fastq files

Tags Genomics NGS Sequences FASTQ Annotation

Purpose

This script iterates over a fastq file and outputs summary statistics for each read.

The output is a tab-delimited text file with the following columns:

Column	*Content*
read	read identifier present in input fastq file
nfailed	number of reads that fall below Q10
nN	number of ambiguous base calls (N)
nval	number of bases in the read
min	minimum base quality score for the read
max	maximum base quality for the read
mean	mean base quality for the read
median	median base quality for the read
stddev	standard devitation of quality scores for the read
sum	sum of quality scores for the read
q1	25th percentile of quality scores for the read
q3	25th percentile of quality scores for the read

Usage

Example:

```
cgat fastq2table --guess-format=sanger < in.fastq > out.tsv
```

In this example we know that our data have quality scores formatted as sanger. Given that illumina-1.8 quality scores are highly overlapping with sanger, this option defaults to sanger qualities. In default mode the script may not be able to distinguish highly overlapping sets of quality scores.

If we provide two reads to the script:

```
@DHKW5DQ1:308:D28FGACXX:5:2211:8051:4398
ACAATGTCCTGATGTGAATGCCCCTACTATTCAGATCGCTTAGGGCATGC
+
B1=?DFDDHHFFHIJJIJGGIJGFIEE9CHIIFEGGIIJGIGIGIIDGHI
@DHKW5DQ1:308:D28FGACXX:5:1315:15039:83265
GAATGCCCCTACTATTCAGATCGCTTAGGGCATGCGTCGCATGTGAGTAA
+
@@@FDFFFHGHHHJIIIJIGHIJJIGHGHC9FBFBGHIIEGHIGC>F@FA
```

we get the following table as output:

read	nfailed	dnN	nval	min	max	mean	median	stddev	sum	q1	q3
DHKW5DQ1:308:D28FGACXX:5:2211:8051:4398	0	0	50	16.0000	41.0000	37.2000	38.0000	4.4900	1860.00	36.0000	40.0000
DHKW5DQ1:308:D28FGACXX:5:1315:15039:83265	0	0	50	24.0000	41.0000	37.0200	38.0000	3.5916	1851.00	36.0000	40.0000

Type:

```
cgat fastq2table --help
```

for command line help.

Command line options

```
usage: fastq2table [-h] [--version]
                   [--guess-format {sanger,solexa,phred64,illumina-1.8,integer}]
                   [--target-format {sanger,solexa,phred64,illumina-1.8,integer}]
                   [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                   [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                   [--log-config-filename LOG_CONFIG_FILENAME]
                   [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                   [-E STDERR] [-S STDOUT]
fastq2table: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
```

(continues on next page)

(continued from previous page)

```
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'filter_reads'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'genelist_analysis'
```

genome_bed.py - Create a bed file tiling a genome from a fai file

Tags Python

This program takes an indexed genome and creates windows of a certain size.

It also takes two input parameters: the window/tile size (bases) and the shift size. By default the shift size is equal to the window size. The default window size is 1000.

Usage

python genome_bed -g <genome.fai> -o <output.bed> -w window size -s shift size

Command line options

```
usage: genome-bed [-h] [--version] [-g GENOME_FILE] [-w WINDOW] [-s SHIFT]
                  [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                  [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                  [--log-config-filename LOG_CONFIG_FILENAME]
                  [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                  [-E STDERR] [-S STDOUT]
genome-bed: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'go2plot'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'gtf2overlap'
```

index2bed.py - convert indexed fasta file to bed file

Tags Python

Purpose

Usage

Type:

```
python <script_name>.py --help
```

for command line help.

Command line options

```
usage: index2bed [-h] [--version] [-g GENOME_FILE]
                 [--remove-regex REMOVE_REGEX] [-e GFF_FILE]
                 [-f FIXED_WIDTH_WINDOWS] [--timeit TIMEIT_FILE]
                 [--timeit-name TIMEIT_NAME] [--timeit-header]
                 [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                 [--log-config-filename LOG_CONFIG_FILENAME]
                 [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                 [-E STDERR] [-S STDOUT]
index2bed: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'intervaltable2bed'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'liftover'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'list_overlap'
```

medip_merge_intervals.py - merge differentially methylated regions

Tags Python

Purpose

This script takes the output of DESeq or EdgeR and merges adjacent intervals that show a similar expression change.

Input is data like this:

```
contig start end treatment_name  treatment_mean  treatment_std   control_name
↪control_mean    control_std     pvalue  qvalue  l2fold  fold    significant
↪status
chr1 10000 11000        CD14    32.9785173324   0       CD4     41.7117152603   0
↪  0.199805206526  1.0     0.338926100945  1.26481475319   0       OK
chr1 14000 15000        CD14    9.32978709019   0       CD4     9.31489982941   0
↪  1.0     1.0     -0.00230390372974       0.998404330063  0       OK
chr1 15000 16000        CD14    9.04603350905   0       CD4     9.01484414416   0
↪  1.0     1.0     -0.00498279072069       0.996552150193  0       OK
chr1 16000 17000        CD14    0.457565479197  0       CD4     0.14910378845   0
↪  0.677265200643  1.0     -1.61766129852  0.325863281276  0       OK
```

The second and third window would be merged, as

1. Their methylation levels are within 10% of each other.
2. They are both not differentially methylated.

It aggregates the following:

- mean values: average
- std values: max

- pvalue: max
- qvalue: max
- fold: min/max (depending on enrichment/depletion)
- l2fold: min/max (depending on enrichment/depletion)

The analysis outputs bed files with intervals that are potentially activated in one of the conditions. Windows with a positive fold change are collected in the `treatment`, while windows with a negative fold change are collected in the `control`.

For methylation analysis, it might be more interesting to report windows that are depleted (instead of enriched) of signal. Thus, if the option `--invert` is given, windows with a negative l2fold change are labeled `treatment`. Less methylation means that this region is "active" in the `treatment` condition.

Note that the input is assumed to be sorted by coordinate.

Usage

Example:

```
python cgat_script_template.py --help
```

Type:

```
python cgat_script_template.py --help
```

for command line help.

Command line options

```
usage: medip-merge-intervals [-h] [--version] [-o MIN_OVERLAP]
                             [-w PATTERN_WINDOW] [-i] [--timeit TIMEIT_FILE]
                             [--timeit-name TIMEIT_NAME] [--timeit-header]
                             [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                             [--log-config-filename LOG_CONFIG_FILENAME]
                             [--tracing {function}] [-? ?]
                             [-P OUTPUT_FILENAME_PATTERN] [-F] [-I STDIN]
                             [-L STDLOG] [-E STDERR] [-S STDOUT]
medip-merge-intervals: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'probeset2gene'
```

cgat_rebuild_extensions.py - rebuild all cython extensions

Tags Python

Purpose

This script rebuilds all cython extensions in the source directory.

Some scripts in the repository make use of `pyximport` to compile associated cython scripts with embedded C code. Theses scripts are automatically re-compiled if the script has changed, but this process can fail if:

- the script is executed on a machine without a C-compiler
- some underlying libraries have changed.

Thus, it is safer to rebuild all scripts on a machine with a C compiler before running a script in production on a cluster, where not all nodes might be fully configured for compilation.

Usage

Example:

```
python cgat_rebuild_extensions.py
```

Type:

```
python cgat_rebuild_extensions.py --help
```

for command line help.

Command line options

```
usage: cgat-rebuild-extensions [-h] [-i TEST_OPTION] [--timeit TIMEIT_FILE]
                               [--timeit-name TIMEIT_NAME] [--timeit-header]
                               [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                               [--log-config-filename LOG_CONFIG_FILENAME]
                               [--tracing {function}] [-? ?] [-I STDIN]
                               [-L STDLOG] [-E STDERR] [-S STDOUT]
cgat-rebuild-extensions: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'revigo'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'snp2snp'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'solexa2stats'
```

vcf2vcf.py - manipulate vcf files

Purpose

Manipulate vcf-formatted files.

Usage

Type:

```
python vcf2vcf.py --help
```

for command line usage.

This script provides the following methods:

re-order
reorder sample columns in vcf formatted file according to a given sort order

`cgat.tools.vcf2vcf.`**`Documentation`**`()`

This is a tool for manipulating vcf-formatted files. The following

options are available:

+-----------+-------------------------+

+-----------+-------------------------+

lift-over

^^^^^^^^^

Command line options

```
usage: vcf2vcf [-h] [--version] [--input-filename-fasta INPUT_FILENAME_FASTA]
               [--input-filename-bam INPUT_FILENAME_BAM]
               [--method {add-strelka-genotype,lift-over}]
               [--input-filename-chain INPUT_FILENAME_CHAIN]
               [--normal-sample-regex NORMAL_SAMPLE_REGEX]
               [--output-filename-unmapped OUTPUT_FILENAME_UNMAPPED]
               [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
               [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
               [--log-config-filename LOG_CONFIG_FILENAME]
               [--tracing {function}] [-? ?] [-P OUTPUT_FILENAME_PATTERN] [-F]
               [-I STDIN] [-L STDLOG] [-E STDERR] [-S STDOUT]
vcf2vcf: error: argument -?: expected one argument
```

vcfstats_sqlite.py - reformat output of vcf-stats for database loading

Tags Python

Purpose

create a csv separated file for loading into a database from output of vcf-stats utility in vcf-tools package.

Usage

Example:

```
python vcfstats_sqlite.py [files] > [outfile]
```

Type:

```
python vcfstats_sqlite.py --help
```

for command line help.

Command line options

```
usage: vcfstats2db [-h] [--version] [--timeit TIMEIT_FILE]
                   [--timeit-name TIMEIT_NAME] [--timeit-header]
                   [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                   [--log-config-filename LOG_CONFIG_FILENAME]
                   [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                   [-E STDERR] [-S STDOUT]
vcfstats2db: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'wig2wig'
```

bam_vs_bam.py - compute coverage correlation between bam files

Tags Genomics NGS BAM Comparison

Purpose

Compare per base coverage between two *bam* formatted files.

Usage

Example:

```
python bam_vs_bam.py in1.bam in2.bam
```

This command generates a tab delimited output with columns chromosome, base coordinate, number of overlapping reads in in1.bam, and number of overlapping reads in in2.bam.

Type:

```
python bam_vs_bam.py --help
```

for command line help.

Documentation

This tools allows users to compare the per base coverage between two BAM files. The output includes all bases in the supplied reference fasta except those with no coverage in the input BAMs.

At present the –interval or -i option has not been implemented.

Command line options

--regex-identifier supply a regex to extract an identifier from the filenames. defualts to using the filename

```
usage: bam-vs-bam [-h] [--version] [-i FILENAME_INTERVALS]
                  [-e REGEX_IDENTIFIER] [--timeit TIMEIT_FILE]
                  [--timeit-name TIMEIT_NAME] [--timeit-header]
                  [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                  [--log-config-filename LOG_CONFIG_FILENAME]
                  [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                  [-E STDERR] [-S STDOUT]
bam-vs-bam: error: argument -?: expected one argument
```

bam_vs_bed.py - count context that reads map to

Tags Genomics NGS Intervals BAM BED Counting

Purpose

This script takes as input a *BAM* file from an RNA-seq or similar experiment and a *bed* formatted file. The *bed* formatted file needs at least four columns. The fourth (name) column is used to group counts.

The script counts the number of alignments overlapping in the first input file that overlap each feature in the second file. Annotations in the *bed* file can be overlapping - they are counted independently.

Note that duplicate intervals will be counted multiple times. This situation can easily arise when building a set of genomic annotations based on a geneset with alternative transcripts. For example:

```
chr1    10000    20000    protein_coding          # gene1, transrcipt1
chr1    10000    20000    protein_coding          # gene1, transcript2
```

Any reads overlapping the interval chr1:10000-20000 will be counted twice into the protein_coding bin by bedtools. To avoid this, remove any duplicates from the *bed* file:

```
zcat input_with_duplicates.bed.gz | cgat bed2bed --merge-by-name | bgzip > input_
↪without_duplicates.bed.gz
```

This scripts requires bedtools to be installed.

Options

-a, –bam-file / -b, –bed-file These are the input files. They can also be provided as provided as positional arguements, with the bam file being first and the (gziped or uncompressed) bed file coming second

-m, --min-overlap Using this option will only count reads if they overlap with a bed entry by a certain minimum fraction of the read.

Example

Example:

```
python bam_vs_bed.py in.bam in.bed.gz
```

Usage

Type:

```
cgat bam_vs_bed BAM BED [OPTIONS]
cgat bam_vs_bed --bam-file=BAM --bed-file=BED [OPTIONS]
```

where BAM is either a bam or bed file and BED is a bed file.

Type:

```
cgat bam_vs_bed --help
```

for command line help.

Command line options

```
usage: bam-vs-bed [-h] [--version] [-m MIN_OVERLAP] [-a bam] [-b bed] [-s]
                  [--assume-sorted] [--split-intervals] [--timeit TIMEIT_FILE]
                  [--timeit-name TIMEIT_NAME] [--timeit-header]
                  [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                  [--log-config-filename LOG_CONFIG_FILENAME]
                  [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                  [-E STDERR] [-S STDOUT]
bam-vs-bed: error: argument -?: expected one argument
```

bam_vs_gtf.py - compare bam file against gene set

Tags Genomics NGS Genesets BAM GTF Summary

Purpose

Compare RNASeq reads in a BAM file and compares it against reference exons to quantify exon overrun / underrun.

Documentation

This script is for validation purposes:

- Exon overrun should be minimal - reads should not extend beyond known exons.
- Spliced reads should link known exons.

Please note:

- For unspliced reads, any bases extending beyond exon boundaries are counted.

- **For spliced reads, both parts of the reads are examined for their overlap.** As a consequence, counts are doubled for spliced reads.
- The script requires a list of non-overlapping exons as input.
- For read counts to be correct the NH (number of hits) flag needs to be set correctly.

Usage

Example:

```
# Preview the BAM file using Samtools view
samtools view tests/bam_vs_gtf.py/small.bam | head
# Pipe input bam to script and specify gtf file as argument
cat tests/bam_vs_gtf.py/small.bam | cgat bam_vs_gtf.py --gtf-file=tests/bam_vs_gtf.py/
↪hg19.chr19.gtf.gz
```

category	counts
spliced_bothoverlap	0
unspliced_overlap	0
unspliced_nooverrun	0
unspliced	207
unspliced_nooverlap	207
spliced_overrun	0
spliced_halfoverlap	0
spliced_exact	0
spliced_inexact	0
unspliced_overrun	0
spliced	18
spliced_underrun	0
mapped	225
unmapped	0
input	225
spliced_nooverlap	18
spliced_ignored	0

Type:

```
python bam_vs_gtf.py --help
```

for command line help.

Command line options

filename-exons / filename-gtf: a gtf formatted file containing the genomic coordinates of a set of non-overlapping exons, such as from a reference genome annotation database (Ensembl, UCSC etc.).

```
usage: bam-vs-gtf [-h] [--version] [-e gtf] [--timeit TIMEIT_FILE]
                  [--timeit-name TIMEIT_NAME] [--timeit-header]
                  [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                  [--log-config-filename LOG_CONFIG_FILENAME]
                  [--tracing {function}] [-? ?] [-P OUTPUT_FILENAME_PATTERN]
```

(continues on next page)

(continued from previous page)

```
                  [-F] [-I STDIN] [-L STDLOG] [-E STDERR] [-S STDOUT]
bam-vs-gtf: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 129, in main
    module = imp.load_module(command, file, pathname, description)
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 235, in load_module
    return load_source(name, filename, file)
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 172, in load_source
    module = _load(spec)
  File "<frozen importlib._bootstrap>", line 696, in _load
  File "<frozen importlib._bootstrap>", line 677, in _load_unlocked
  File "<frozen importlib._bootstrap_external>", line 728, in exec_module
  File "<frozen importlib._bootstrap>", line 219, in _call_with_frames_removed
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/tools/cgat2rdf.py", line 81, in <module>
    from rdflib import Graph
ModuleNotFoundError: No module named 'rdflib'
```

diff_bam.py - compare multiple bam files against each other

Tags Genomics NGS BAM Comparison

Purpose

Compare reads in multiple BAM files against each other.

Note: BAM files need to be sorted by read name. samtools sort does NOT work as it uses a custom comparison function (strnum_cmp) that is incompatible with the standard lexicographical order in python. See the example below on how to get sorted files.

This script is for validation purposes. It might take a while for large BAM files.

Usage

If you have two sorted *sam* or *bam* formatted files, type:

```
cgat diff_bam a.bam b.bam > out
```

If they are not sorted, you can use samtools sort to do an inplace sort:

```
cgat diff_bam <(samtools view -h a.bam | hsort 0 -k1,1)
              <(samtools view -h b.bam | hsort 0 -k1,1)
```

The samtools `-h` option outputs the header, and the hsort command sorts without disturbing the header.

An example output looks like this:

read	nloca-tions	nmatched	file1_nh	file2_nh	file1_loc	file2_loc
42YKVAAXX_HWI-EAS229_1:1:11:1659:174	1	2	2	2	0,0	0,0
42YKVAAXX_HWI-EAS229_1:1:11:166:1768	1	2	1	1	0	0
612UOAAXX_HWI-EAS229_1:1:97:147:1248	2	2	2	2	0,1	0,1

This reports for each read the number of locations that the read maps to in all files, the number of files that have matches found for the read. Then, for each file, it reports the number of matches and the locations it maps to (coded as integers, 0 the first location, 1 the second, . . .).

In the example above, the first read maps twice to 1 location in both files. This is a read occuring twice in the input file. The second read maps to the same one location in both files, while the third read maps to the two same locations in both input files.

Type:

```
python diff_bam.py --help
```

for command line help.

Documentation

For read counts to be correct the NH flag to be set correctly.

Command line options

```
usage: diff-bam [-h] [--version] [--header-names HEADERS]
                [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                [--log-config-filename LOG_CONFIG_FILENAME]
                [--tracing {function}] [-? ?] [-P OUTPUT_FILENAME_PATTERN]
                [-F] [-I STDIN] [-L STDLOG] [-E STDERR] [-S STDOUT]
diff-bam: error: argument -?: expected one argument
```

fasta2fasta.py - operate on sequences

Tags Sequences

Purpose

perform operations (masking, renaming) on a stream of fasta formatted sequences.

Available edit operations are:

translate translate sequences using the standard genetic code.

translate-to-stop translate until first stop codon

truncate-at-stop truncate sequence at first stop codon

back-translate convert nucleotide sequence to peptide sequence Requires parameter of second fasta file with peptide sequences.

mark-codons adds a space after each codon

apply-map rename sequence identifiers from a given map Requires parameter with filename of a map. The map is a tab-separated file mapping old to new names.

build-map rename sequence identifiers numerically and save output in a tab-separated file. Requires parameter with filename of a map. The map is a tab-separated file mapping new to old names and will be newly created. Any exiting file of the same name will be overwritten.

pseudo-codons translate, but keep register with codons

interleaved-codons mix amino acids and codons

filter remove sequence according to certain criteria. For example, –method=filter –filter-method=min-length=5 –filter-method=max-length=10

map-codons:

remove-gaps remove all gaps in the sequence

mask-stops mask all stop codons

mask-seg mask sequence by running seg

mask-bias mask sequence by running bias

mask-codons mask codon sequence given a masked amino acid sequence. Requires parameter with masked amino acids in fasta format.

mask-incomplete-codons mask codons that are partially masked or gapped

mask-soft combine hard-masked (NNN) sequences with unmasked sequences to generate soft masked sequence (masked regions in lower case)

remove-stops remove stop codons

upper convert sequence to upper case

lower convert sequence to lower case

reverse-complement build the reverse complement

shuffle shuffle each sequence

sample select a certain proportion of sequences

Parameters are given to the option `parameters` in a comma-separated list in the order that the edit operations are called upon.

Exclusion/inclusion is tested before applying any id mapping.

Usage

Example:

```
python fasta2fasta.py --method=translate < in.fasta > out.fasta
```

Type:

```
python fasta2fasta.py --help
```

for command line help.

Command line options

```
usage: fasta2fasta [-h] [--version]
                   [-m {translate,translate-to-stop,truncate-at-stop,back-translate,
↪mark-codons,apply-map,build-map,pseudo-codons,filter,interleaved-codons,map-codons,
↪remove-gaps,mask-seg,mask-bias,mask-codons,mask-incomplete-codons,mask-stops,mask-
↪soft,map-identifier,nop,remove-stops,upper,lower,reverse-complement,sample,shuffle}]
                   [-p PARAMETERS] [-x]
                   [--sample-proportion SAMPLE_PROPORTION]
                   [--exclude-pattern EXCLUDE_PATTERN]
                   [--include-pattern INCLUDE_PATTERN]
                   [--filter-method FILTER_METHODS] [-t {aa,na}]
                   [-l TEMPLATE_IDENTIFIER] [--map-tsv-file MAP_TSV_FILE]
                   [--fold-width FOLD_WIDTH] [--timeit TIMEIT_FILE]
                   [--timeit-name TIMEIT_NAME] [--timeit-header]
                   [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                   [--log-config-filename LOG_CONFIG_FILENAME]
                   [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                   [-E STDERR] [-S STDOUT]
fasta2fasta: error: argument -?: expected one argument
```

fasta2kmercontent.py

Tags Genomics Sequences FASTA Summary

Purpose

This script takes an input *fasta* file from stdin and computes a k-nucleotide content for each contig in the file. The output is a tab-delimited file of kmer counts:

```
     contig1  contig2  contig3  contig4
n1
n2
n3
```

where n is the kmer and contig is the fasta entry.

The user specifies the kmer length that is to be searched. Note that the longer the kmer, the longer the script will take to run.

Note the order of output will not necessarily be the same order as the input.

Usage

Example:

```
zcat in.fasta.gz | head::

 >NODE_1_length_120_cov_4.233333
 TCACGAGCACCGCTATTATCAGCAACTTTTAAGCGACTTTCTTGTTGAATCATTTCAATT
 GTCTCCTTTTAGTTTTATTAGATAATAACAGCTTCTTCCACAACTTCTACAAGACGGAAG
 CGTTTTGTAGCTGAAAGTGGGCGAGTTTCCATGATACGAAcgatATCGCC

 >NODE_3_length_51_cov_33.000000
 CGAGTTTCCATGATACGAAcgatATCGCCTTCTTTAGCAACGTTGTTTTCGTCATGTGCT
 TTATATTTTTTAGAATAGTTGATACGTTTACCATAGACTGG

zcat in.fasta.gz | python fasta2kmercontent.py
                   --kmer-size 4
                   > tetranucleotide_counts.tsv

head tetranucleotide_counts.tsv::

  kmer NODE_228_length_74_cov_506.432434 NODE_167_length_57_cov_138.438599
  GTAC 0                                 0
  TGCT 0                                 0
  GTAA 2                                 0
  CGAA 1                                 1
  AAAT 1                                 0
  CGAC 0                                 0
```

In this example, for each contig in in.fasta.gz the occurrence of each four nucleotide combination is counted.

Alternative example:

```
zcat in.fasta.gz | python fasta2kmercontent.py
                   --kmer-size 4
                   --output-proportion
                   > tetranucleotide_proportions.tsv
```

In this example, for each contig in in.fasta.gz we return the proportion of each four base combination out of the total tetranucleotide occurences. `--output-proportion` overides the count output.

Options

Two options control the behaviour of fasta2kmercontent.py; `--kmer-size` and `--output-proportion`.

`--kmer-size::` The kmer length to count over in the input fasta file

`--output-proportion::` The output values are proportions rather than absolute counts

Type:

```
python fasta2composition.py --help
```

for command line help.

Command line options

```
usage: fasta2kmercontent [-h] [--version] [-k KMER] [-p]
                         [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                         [--timeit-header] [--random-seed RANDOM_SEED]
                         [-v LOGLEVEL]
                         [--log-config-filename LOG_CONFIG_FILENAME]
                         [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                         [-E STDERR] [-S STDOUT]
fasta2kmercontent: error: argument -?: expected one argument
```

fastas2fasta.py - concatenate sequences from multiple fasta files

Tags Genomics Sequences MultipleAlignments FASTA Manipulation

Purpose

This script reads sequences from two or more *fasta* formatted files and outputs a new file with the sequences concatenated per entry.

All files must have the same number of sequences and the id of the first file is output.

Usage

Example:

```
python fastas2fasta.py a.fasta b.fasta > c.fasta
```

If a.fasta is:

```
>1
AAACC
>2
CCCAA
```

and b.fasta is:

```
>a
GGGGTTT
>b
TTTTGGG
```

then the output will be:

```
>1
AAACCGGGGTTT
>2
CCCAATTTTGGG
```

Type:

```
python fastas2fasta.py --help
```

for command line help.

Command line options

```
usage: fastas2fasta [-h] [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                    [--timeit-header] [--random-seed RANDOM_SEED]
                    [-v LOGLEVEL] [--log-config-filename LOG_CONFIG_FILENAME]
                    [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                    [-E STDERR] [-S STDOUT]
fastas2fasta: error: argument -?: expected one argument
```

fastqs2fasta.py - interleave two fastq files

Tags Genomics NGS FASTQ FASTA Conversion

Purpose

This script is used to interleave two *fastq*-formatted files (paired data) into a single *fasta*-formatted file. Read1 is followed by read2 in the resultant file.

fastq files MUST be sorted by read identifier.

Usage

For example:

```
cgat fastqs2fasta          --first-fastq-file=in.fastq.1.gz          --second-fastq-
↪file=in.fastq.2.gz > out.fasta
```

If `in.fastq.1.gz` looks like this:

```
@r1_from_gi|387760314|ref|NC_017594.1|_Streptococcus_saliva_#0/1
TTCTTGTTGAATCATTTCAATTGTCTCCTTTTAGTTTTATTAGATAATAACAGCTTCTTCCACAACTTCT
+
??A???ABBDDBDDEDGGFGAFHHCHHIIIDIHGIFIH=HFICIHDHIHIFIFIIIIIIHFHIFHIHHHH
@r3_from_gi|315441696|ref|NC_014814.1|_Mycobacterium_gilvum_#0/1
ATGAACGCGGCCGAGCAACACCGCCACCACGTGAATCGGTGGTTCTACGACTGCCCGTCGGCCTTCCACC
+
```

and `in.fastq.2.gz` looks like this:

```
A??A?B??BDBDDDBDGGFA>CFCFIIIIIIF;HFIGHCIGHIHHEHHHIIHHFDHH-HD-IDHHHGIHG
@r1_from_gi|387760314|ref|NC_017594.1|_Streptococcus_saliva_#0/2
ACCTTCGTTTCCAAGGTGCAGCAGGTCAACTTGATCAAACTGCCCCTTTGAACGAAGTGAAAAAACAAAT
+
A????@BBDBDDADABGFGFFEHHHIEHHII@IIHIHHIDHCCIHIIIHHIEI5HIHFHIEHIH=CHHC)
@r3_from_gi|315441696|ref|NC_014814.1|_Mycobacterium_gilvum_#0/2
GGGAGCCTGCAGCGCCGCCGCGACTGCATCGCCGCGGCCGGCATCGTGGGATGGACGGTGCGTCAGACGC
+
???A?9BBDDD5@DDDGFFGFFHIIIHHIHBFHIIHIIHHH>HEIHHFI>FFHGIIHHHDHCCFIHFIHD
```

then the output will be:

```
>r1_from_gi|387760314|ref|NC_017594.1|_Streptococcus_saliva_#0/1
TTCTTGTTGAATCATTTCAATTGTCTCCTTTTAGTTTTATTAGATAATAACAGCTTCTTCCACAACTTCT
>r1_from_gi|387760314|ref|NC_017594.1|_Streptococcus_saliva_#0/2
ACCTTCGTTTCCAAGGTGCAGCAGGTCAACTTGATCAAACTGCCCCTTTGAACGAAGTGAAAAAACAAAT
>r3_from_gi|315441696|ref|NC_014814.1|_Mycobacterium_gilvum_#0/1
ATGAACGCGGCCGAGCAACACCGCCACCACGTGAATCGGTGGTTCTACGACTGCCCGTCGGCCTTCCACC
>r3_from_gi|315441696|ref|NC_014814.1|_Mycobacterium_gilvum_#0/2
GGGAGCCTGCAGCGCCGCCGCGACTGCATCGCCGCGGCCGGCATCGTGGGATGGACGGTGCGTCAGACGC
>r4_from_gi|53711291|ref|NC_006347.1|_Bacteroides_fragilis_#0/1
GAGGGATCAGCCTGTTATCCCCGGAGTACCTTTTATCCTTTGAGcgatGTCCCTTCCATACGGAAACACC
>r4_from_gi|53711291|ref|NC_006347.1|_Bacteroides_fragilis_#0/2
CAACCGTGAGCTCAGTGAAATTGTAGTATCGGTGAAGATGCcgatTACCCGcgatGGGACGAAAAGACCC
>r5_from_gi|325297172|ref|NC_015164.1|_Bacteroides_salanitr_#0/1
TGCGGCGAAATACCAGCCCATGCCCCGTCCCCAGAATTCCTTGGAGCAGCCTTTGTGAGGTTCGGCTTTG
>r5_from_gi|325297172|ref|NC_015164.1|_Bacteroides_salanitr_#0/2
AACGGCACGCACAATGCCGACCGCTACAAAAAGGCTGCCGACTGGCTCCGCAATTACCTGGTGAACGACT
```

Type:

```
cgat fastqs2fasta --help
```

for command line help.

Command line options

```
usage: fastqs2fasta [-h] [--version] [-a FASTQ1] [-b FASTQ2]
                    [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                    [--timeit-header] [--random-seed RANDOM_SEED]
                    [-v LOGLEVEL] [--log-config-filename LOG_CONFIG_FILENAME]
                    [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                    [-E STDERR] [-S STDOUT]
fastqs2fasta: error: argument -?: expected one argument
```

fastqs2fastqs.py - manipulate (merge/reconcile) fastq files

Tags Genomics NGS FASTQ FASTQ Manipulation

Purpose

This script manipulates multiple fastq files and outputs new fastq files. Currently only the method `reconcile` is implemented.

reconcile

Reconcile reads from a pair of fastq files.

This method takes two fastq files and outputs two fastq files such that all reads in the output are present in both output files.

The typical use case is that two fastq files containing the first and second part of a read pair have been independently filtered, for example by quality scores, truncation, etc. As a consequence some reads might be missing from one file but not the other. The reconcile method will output two files containing only reads that are common to both files.

The two files must be sorted by read identifier.

Example input, read2 and read3 are only present in either of the files:

> # File1 # File 2
>
> @read1 @read1 AAA AAA + + !!! !!! @read2 @read3 CCC TTT + + !!! !!! @read4 @read4 GGG GGG + + !!! !!!

Example output, only the reads common to both files are output:

```
# File1         # File 2

@read1          @read1
AAA             AAA
+               +
!!!             !!!
@read4          @read4
GGG             GGG
+               +
!!!             !!!
```

Usage

Example:

```
python fastqs2fastqs.py             --method=reconcile             --output-filename-
↪pattern=myReads_reconciled.%s.fastq            myReads.1.fastq.gz myReads.2.fastq.
↪gz
```

In this example we take a pair of fastq files, reconcile by read identifier and output 2 new fastq files named `myReads_reconciled.1.fastq.gz` and `myReads_reconciled.2.fastq.gz`.

Type:

```
python fastqs2fastqs.py --help
```

for command line help.

Command line options

```
usage: fastqs2fastqs [-h] [--version] [-m {reconcile,filter-by-sequence}] [-c]
                     [-u] [--id-pattern-1 ID_PATTERN_1]
                     [--id-pattern-2 ID_PATTERN_2]
                     [--input-filename-fasta INPUT_FILENAME_FASTA]
                     [--filtering-kmer-size FILTERING_KMER_SIZE]
                     [--filtering-min-kmer-matches FILTERING_MIN_KMER_MATCHES]
                     [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                     [--timeit-header] [--random-seed RANDOM_SEED]
                     [-v LOGLEVEL] [--log-config-filename LOG_CONFIG_FILENAME]
                     [--tracing {function}] [-? ?]
                     [-P OUTPUT_FILENAME_PATTERN] [-F] [-I STDIN] [-L STDLOG]
                     [-E STDERR] [-S STDOUT]
fastqs2fastqs: error: argument -?: expected one argument
```

gtf2tsv.py - convert gtf file to a tab-separated table

Tags Genomics Genesets

Purpose

convert a gtf formatted file to tab-separated table. The difference to a plain *gtf* formatted file is that column headers are added, which can be useful when importing the gene models into a database.

Note that coordinates are converted to 0-based open/closed notation (all on the forward strand).

By default, the gene_id and transcript_id are extracted from the attributes field into separated columns. If `-f/--attributes-as-columns` is set, all fields in the attributes will be split into separate columns.

The script also implements the reverse operation, converting a tab-separated table into a *gtf* formatted file.

When using the `-m, --map` option, the script will output a table mapping gene identifiers to transcripts or peptides.

USING GFF3 FILE: The script also can convert gff3 formatted files to tsv files when specifiying the option –is-gff3 and –attributes-as-columns. Currently only the full GFF3 to task is implimented. Further improvements to this script can be made to only output the attributes only, i.e. –output-only-attributes.

Usage

Example:

```
cgat gtf2tsv < in.gtf
```

contig	source	feature	start	end	score	strand	frame	gene_id	transcript_id	attributes
chr19	processed_transcript	exon	66345	66509	.	+	.	ENSG00000233750	ENST00000592209	exon_number "1"; gene_name "AC008993.5"; gene_biotype "pseudogene"; transcript_name "AC008993.5-002"; exon_id "ENSE00001701708"
chr19	processed_transcript	exon	60520	60747	.	+	.	ENSG00000233750	ENST00000592209	exon_number "2"; gene_name "AC008993.5"; gene_biotype "pseudogene"; transcript_name "AC008993.5-002"; exon_id "ENSE00002735807"
chr19	processed_transcript	exon	60104	60162	.	+	.	ENSG00000233750	ENST00000592209	exon_number "3"; gene_name "AC008993.5"; gene_biotype "pseudogene"; transcript_name "AC008993.5-002"; exon_id "ENSE00002846866"

To build a map between gene and transcrip identiers, type:

```
cgat gtf2tsv --output-map=transcript2gene < in.gtf
```

transcript_id	gene_id
ENST00000269812	ENSG00000141934
ENST00000318050	ENSG00000176695
ENST00000327790	ENSG00000141934

To run the script to convert a gff3 formatted file to tsv, type:

```
cat file.gff3.gz | cgat gtf3tsv --is-gff3 --attributes-as-columns
> outfile.tsv
```

Type:

```
cgat gtf2tsv --help
```

for command line help.

Command line options

```
usage: gtf2tsv [-h] [--version] [-o] [-f] [--is-gff3] [-i]
               [-m {transcript2gene,peptide2gene,peptide2transcript}]
               [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
               [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
               [--log-config-filename LOG_CONFIG_FILENAME]
               [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
               [-E STDERR] [-S STDOUT]
gtf2tsv: error: argument -?: expected one argument
```

gtfs2tsv.py - compare two genesets

Tags Python

Purpose

This script compares two genesets (required) in *gtf*-formatted files and output lists of shared and unique genes.

It outputs the results of the comparison into various sections. The sections are split into separate output files whose names are determined by the `--output-filename-pattern` option. The sections are:

genes_ovl Table with overlapping genes

genes_total Summary statistic of overlapping genes

genes_uniq1 List of genes unique in set 1

genes_uniq2 List of genes unique in set 2

Options

--output-filename-pattern This option defines how the output filenames are determined for the sections described in the Purpose section above.

Usage

Example:

```
head a.gtf::

  19 processed_transcript exon 66346 66509 . - . gene_id "ENSG00000225373";
  transcript_id "ENST00000592209"; exon_number "1"; gene_name "AC008993.5";
  gene_biotype "pseudogene"; transcript_name "AC008993.5-002";
  exon_id "ENSE00001701708";

  19 processed_transcript exon 60521 60747 . - . gene_id "ENSG00000225373";
  transcript_id "ENST00000592209"; exon_number "2"; gene_name "AC008993.5";
  gene_biotype "pseudogene"; transcript_name "AC008993.5-002";
  exon_id "ENSE00002735807";

  19 processed_transcript exon 60105 60162 . - . gene_id "ENSG00000225373";
  transcript_id "ENST00000592209"; exon_number "3"; gene_name "AC008993.5";
  gene_biotype "pseudogene"; transcript_name "AC008993.5-002";
  exon_id "ENSE00002846866";

head b.gtf::

  19 transcribed_processed_pseudogene exon 66320 66492 . - .
  gene_id "ENSG00000225373"; transcript_id "ENST00000587045"; exon_number "1";
  gene_name "AC008993.5"; gene_biotype "pseudogene";
  transcript_name "AC008993.5-001"; exon_id "ENSE00002739353";

  19 lincRNA exon 68403 69146 . + . gene_id "ENSG00000267111";
  transcript_id "ENST00000589495"; exon_number "1"; gene_name "AC008993.2";
  gene_biotype "lincRNA"; transcript_name "AC008993.2-001";
  exon_id "ENSE00002777656";

  19 lincRNA exon 71161 71646 . + . gene_id "ENSG00000267588";
  transcript_id "ENST00000590978"; exon_number "1"; gene_name "MIR1302-2";
  gene_biotype "lincRNA"; transcript_name "MIR1302-2-001";
  exon_id "ENSE00002870487";

python gtfs2tsv.py a.gtf b.gtf > out.tsv

head out.tsv::

  contigs source feature start end score strand frame gene_id transcript_id attributes
  19 processed_transcript exon 66345 66509 . - . ENSG00000225373 ENST00000592209 exon_
↪number "1";
  gene_name "AC008993.5"; gene_biotype "pseudogene"; transcript_name "AC008993.5-002";
  exon_id "ENSE00001701708"
  19 processed_transcript exon 60520 60747 . - . ENSG00000225373 ENST00000592209 exon_
↪number "2";
  gene_name "AC008993.5"; gene_biotype "pseudogene"; transcript_name "AC008993.5-002";
  exon_id "ENSE00002735807"
```

Type:

```
python gtfs2tsv.py --help
```

for command line help.

Command line options

```
usage: gtfs2tsv [-h] [--version] [-e] [-f] [-p] [-s] [--timeit TIMEIT_FILE]
                [--timeit-name TIMEIT_NAME] [--timeit-header]
                [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                [--log-config-filename LOG_CONFIG_FILENAME]
                [--tracing {function}] [-? ?] [-P OUTPUT_FILENAME_PATTERN]
                [-F] [-I STDIN] [-L STDLOG] [-E STDERR] [-S STDOUT]
gtfs2tsv: error: argument -?: expected one argument
```

rnaseq_junction_bams2bam.py - convert mappings against junctions to genomic coordinates

Tags Genomics NGS Genesets

Purpose

This script takes as input a BAM file resulting from reads mapped against a junction database and outputs a *bam* formatted file in genomic coordinates.

The contigs should be of the format <chromosome>|<start>|<exon-end>-<exon-start>|<end>|<splice>|<strand>.

<start> - 0-based coordinate of first base <exon-end> - 0-based coordinate of last base in exon <exon-start> - 0-based coordinate of first base in exon <end> - 0-based coordinate of base after last base

Strand can be either `fwd` or `rev`, though sequences in the database and coordinates are all on the forward strand.

For example `chr1|1244933|1244982-1245060|1245110|GTAG|fwd` translates to the intron `chr1:1244983-1245060` in python coordinates.

The input bam-file is supposed to be sorted by read. Only the best matches are output for each read, were best is defined both in terms of number of mismatches and number of colour mismatches.

Usage

Example:

```
cat input.bam | python rnaseq_junction_bam2bam.py - --log=log > output.bam
```

Type:

```
python rnaseq_junction_bam2bam.py --help
```

for command line help.

Command line options

```
usage: rnaseq-junction-bam2bam [-h] [--version] [-t FILENAME_GENOME_BAM]
                               [-s FILENAME_CONTIGS] [-o] [-i]
                               [-c REMOVE_CONTIGS] [-f] [-u]
                               [--timeit TIMEIT_FILE]
                               [--timeit-name TIMEIT_NAME] [--timeit-header]
                               [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                               [--log-config-filename LOG_CONFIG_FILENAME]
                               [--tracing {function}] [-? ?] [-I STDIN]
                               [-L STDLOG] [-E STDERR] [-S STDOUT]
rnaseq-junction-bam2bam: error: argument -?: expected one argument
```

split_gff - split a gff file into chunks

Tags Genomics Intervals Genesets GFF Manipulation

Purpose

Split gff file into chunks. Overlapping entries will always be output in the same chunk. Input is read from stdin unless otherwise specified. The input needs to be contig/start position sorted.

Options

-i –min-chunk-size

This option specifies how big each chunck should be, in terms of the number of gff lines to be included. Because overlapping lines are always output to the same file, this should be considered a minimum size.

-n, --dry-run This options tells the script not to actaully write any files, but it will output a list of the files that would be output.

Example

cgat splitgff -i 1 < in.gff

where in.gff looks like:

chr1 . exon 1 10 . + . chr1 . exon 8 100 . + . chr1 . exon 102 150 . + .

will produce two files that look like:

000001.chunk: chr1 . exon 1 10 . + . chr1 . exon 8 100 . + .

000002.chunk: chr1 . exon 102 150 . + .

Usage

cgat splitgff [OPTIONS]

Will read a gff file from stdin and split into multiple gff files.

cgat split_gff -I GFF [OPTIONS]

Will read the gff file GFF and split into multiple gff files.

Command line options

```
usage: split-gff [-h] [-i MIN_CHUNK_SIZE] [-n]
                 [--output-filename-name OUTPUT_FILENAME_NAME]
                 [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                 [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                 [--log-config-filename LOG_CONFIG_FILENAME]
                 [--tracing {function}] [-? ?] [-P OUTPUT_FILENAME_PATTERN]
                 [-F] [-I STDIN] [-L STDLOG] [-E STDERR] [-S STDOUT]
split-gff: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'annotator_distance'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'bam2bidirectionaltranscription'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'bam2profile'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'bam2species_map'
```

bams2bam.py - merge genomic and transcriptome mapped bamfiles

Tags Genomics NGS Geneset BAM Manipulation

Purpose

This script takes as input two BAM files from an RNASeq experiment. The first bam file (`bamG`) should contain reads mapped against the genome using a mapper permitting splicing (e.g. tophat). The second bam file (`bamT`) should contain reads mapped against known transcripts. This script will write a new bam file that removes reads from *bamG* that map to regions that are conflicting with those in *bamT*.

Note: Note that if junctions are supplied, the resultant bam files will not be sorted by position.

bamG *bam* formatted file with reads mapped against the genome

bamT *bam* formatted file with reads mapped against transcripts

Usage

Example:

```
python bams2bam.py bamT.bam bamG.bam
```

Type:

```
python bams2bam.py --help
```

for command line help.

Documentation

The script needs to look-up reads via their names. It thus builds an index of reads mapping

This script requires the NM attributes to be set. If it is not set, you will need to set a policy.

Command line options

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 132, in main
    module.main(sys.argv)
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/tools/bams2bam.py", line 79, in main
    usage=globals()["__doc__"])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgatcore/experiment.py", line 510, in __init__
    **kwargs)
TypeError: __init__() got an unexpected keyword argument 'version'
```

bed.plot.py - create genomic snapshots using the IGV Viewer

Tags Python

Purpose

Create genomic plots in a set of intervals using the IGV snapshot mechanism.

The script can use a running instance of IGV identified by host and port. Alternatively, it can start IGV and load a pre-built session.

Usage

Example:

```
python bed2plot.py < in.bed
```

Type:

```
python script_template.py --help
```

for command line help.

Command line options

```
usage: bed2plot [-h] [-s SESSION] [-d SNAPSHOTDIR] [-f {png,eps,svg}]
                [-o HOST] [-p PORT] [-e EXTEND] [-x EXPAND] [--session-only]
                [-n {bed-name,increment}] [--timeit TIMEIT_FILE]
                [--timeit-name TIMEIT_NAME] [--timeit-header]
                [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                [--log-config-filename LOG_CONFIG_FILENAME]
                [--tracing {function}] [-? ?] [-P OUTPUT_FILENAME_PATTERN]
                [-F] [-I STDIN] [-L STDLOG] [-E STDERR] [-S STDOUT]
bed2plot: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'bigwig2hilbert'
```

```
cgat.py - Computational Genomics Analysis Tools
================================================

:Tags: Genomics

To use a specific tool, type::

    cgat <tool> [tool options] [tool arguments]

Tools are grouped by keywords. For this message and a list of
available keywords type::

    cgat --help

For a list of tools matching a certain keyword, type::

   cgat --help <keyword>

or::

   cgat --help all

for a list of all available tools.

To get help for a specific tool, type::

    cgat <tool> --help

cgat tools are grouped by keywords. The following keywords
are defined:

Genomics              FASTQ                 Annotation
```

(continues on next page)

(continued from previous page)

```
NGS                 GenomeAlignment     MultipleAlignments
Geneset             PSL                 Counting
BAM                 CHAIN               Fasta
Manipulation        Summary             Variants
Intervals           Comparison          Protein
BED                 FASTA               WIGGLE
GFF                 Sequences           BIGWIG
Conversion          Genesets            BEDGRAPH
Python              GTF
```

cgat2dot.py - create a graph between cgat scripts

Tags Python

Purpose

This script creates an rdf description of a cgat script.

Optionally, the script outputs also a galaxy xml description of the scripts' interface.

Usage

Example:

```
python cgat2dot.py scripts/*.py
```

Type:

```
python cgat2dot.py --help
```

for command line help.

Documentation

Command line options

```
usage: cgat2dot [-h] [-f {rdf,galaxy}] [-l FILENAME_LIST] [-s SRC_DIR]
                [-r INPUT_REGEX] [-p OUTPUT_PATTERN] [--timeit TIMEIT_FILE]
                [--timeit-name TIMEIT_NAME] [--timeit-header]
                [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                [--log-config-filename LOG_CONFIG_FILENAME]
                [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                [-E STDERR] [-S STDOUT]
cgat2dot: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
```

(continues on next page)

(continued from previous page)

```
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'cgat_add_preamble'
```

```

```

cgat_get_options.py - build a sorted list of all options used in scripts

Author

Tags Python

Purpose

Go through all scripts in the cgat code collection and collect options used in the scripts.

This script expects to be executed at the root of the cgat code repository.

Usage

Example:

```
python cgat_get_options.py
```

Type:

```
python cgat_get_options.py --help
```

for command line help.

Command line options

```
usage: cgat-get-options [-h] [--inplace] [--options-tsv-file TSV_FILE]
                        [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                        [--timeit-header] [--random-seed RANDOM_SEED]
                        [-v LOGLEVEL]
                        [--log-config-filename LOG_CONFIG_FILENAME]
                        [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                        [-E STDERR] [-S STDOUT]
cgat-get-options: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
```

(continues on next page)

(continued from previous page)

```
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'cgat_list_dependencies'
```

cgat_pep8_check_code_quality.py - check PEP8 conformance of cgat Code

Author

Tags Python

Purpose

This script runs pep8.py on the cgat code collection and outputs summary statistics of code quality onto stdout.

Usage

To use, simply run the script from the root directory of the cgat code collection:

```
python cgat_pep8_check_code_quality.py
```

Type:

```
python cgat_pep8_check_code_quality.py --help
```

for command line help.

Command line options

```
usage: cgat-pep8-code-quality [-h] [--timeit TIMEIT_FILE]
                              [--timeit-name TIMEIT_NAME] [--timeit-header]
                              [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                              [--log-config-filename LOG_CONFIG_FILENAME]
                              [--tracing {function}] [-? ?] [-I STDIN]
                              [-L STDLOG] [-E STDERR] [-S STDOUT]
cgat-pep8-code-quality: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'cgat_scan_email'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'clusters2metrics'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'combine_files'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'contigs2random_sample'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'contigs2stats'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
```

(continues on next page)

(continued from previous page)

```
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'counts2counts'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'counts2table'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'coverage2stats'
```

csv_select.py - select rows from a table

Tags Python

Purpose

extract rows from a csv-formatted table.

The select statement is a one-line, for example:

```
csv_select.py "int(r['mC-foetal-sal-R4']) > 0" < in > out
```

Note the required variable name r for denoting field names. Please also be aware than numeric values need to be converted first before testing.

Usage

Type:

```
python csv_select.py --help
```

for command line help.

Command line options

```
usage: csv-select [-h] [-r] [-u] [-l] [-f FILENAME_FIELDS]
                  [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                  [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                  [--log-config-filename LOG_CONFIG_FILENAME]
                  [--tracing {function}] [-? ?] [--csv-dialect CSV_DIALECT]
                  [-I STDIN] [-L STDLOG] [-E STDERR] [-S STDOUT]
csv-select: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'data2resamples'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'data2spike'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'diff_transcript_sets'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
```

(continues on next page)

(continued from previous page)

```
ImportError: No module named 'diffgene2venn'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'distance2clusters'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'distance2merge'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'ena2table'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'expression2distance'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
```

(continues on next page)

(continued from previous page)

```
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'expression2expression'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'fasta2distances'
```

fastq2summary.py - compute summary stats for a fastq file

Tags Genomics NGS Sequences FASTQ Annotation

Purpose

This script iterates over a fastq file and outputs summary statistics for the complete file

The output is a tab-delimited text file with the some of following columns depending on the option specified:

Column	*Content*
reads	total reads in file
bases	total bases in file
mean_length	mean read length
median_length	median read length
mean_quality	mean read quality
median_quality	median read quality
nfailed	number of bases below quality threshold

Usage

Example:

```
python fastq2summary.py --guess-format=sanger < in.fastq > out.tsv
```

In this example we know that our data have quality scores formatted as sanger. Given that illumina-1.8 quality scores are highly overlapping with sanger, this option defaults to sanger qualities. In default mode the script may not be able to distinguish highly overlapping sets of quality scores.

Type:

```
python fastq2summary.py --help
```

for command line help.

Command line options

```
usage: fastq2summary [-h]
                     [--guess-format {sanger,solexa,phred64,illumina-1.8,integer}]
                     [-f {sanger,solexa,phred64,illumina-1.8,integer}]
                     [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                     [--timeit-header] [--random-seed RANDOM_SEED]
                     [-v LOGLEVEL] [--log-config-filename LOG_CONFIG_FILENAME]
                     [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                     [-E STDERR] [-S STDOUT]
fastq2summary: error: argument -?: expected one argument
```

fastqs2fastq.py - merge reads in fastq files

Tags Genomics NGS FASTQ FASTQ Manipulation

Purpose

This script takes two paired-ended fastq files and outputs a single fastq file in which reads have merged.

The two files must be sorted by read identifier.

Note that this script is currently a proof-of-principle implementation and has not been optimized for speed or functionality.

Usage

Example:

```
python fastqs2fastq.py myReads.1.fastq.gz myReads.2.fastq.gz
       --method=join
       > join.fastq
```

In this example we take a pair of fastq files, join the reads and save the output in `join.fastq`.

Type:

```
python fastqs2fastq.py --help
```

for command line help.

Command line options

```
usage: fastqs2fastq [-h] [-m {join}] [--timeit TIMEIT_FILE]
                    [--timeit-name TIMEIT_NAME] [--timeit-header]
                    [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                    [--log-config-filename LOG_CONFIG_FILENAME]
                    [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                    [-E STDERR] [-S STDOUT]
fastqs2fastq: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'formatMetagenemark'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'genes2genes'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'geo2table'
```

gff32gtf.py - various methods for converting gff3 files to gtf

Tags Python

Purpose

Provide a range of methods for converting GFF3 formated files to valid GTF format files.

Background

While the various flavours of GFF format are supposedly backward compatible, this is broken by GTF2.2 and GFF3. GTF requires the presence of gene_id and transcript_id fields for each record. This not so for GFF3. Further key,value tags in the attributes fields of GTF are " " delimited, but are "=" delimited in GFF.

Conversion is non-trivial. GFF3 records are hierachical. To find the gene_id and transcript_id one must traverse the hierarchy to the correct point. Futher records can have multiple parents.

-> Exon

While the standard structure is Gene -> mRNA -| , -> CDS

this is not manditory, and it is possible the conversion will want to be done in a different way.

Usage

Example:

```
python gff32gtf.py --method=[METHOD] [options]
```

Their are several ways in which the conversion can be done:

hierachical

By default this script will read in the entire GFF3 file, and then for each entry traverse the hierarchy until an object of type GENE_TYPE ("gene" by default") or an object with no parent is found. This becomes the "gene_id". Any object of TRANSCRIPT_TYPE encountered on the way is set as the transcript_id. If not such object is encountered then the object directly below the gene object is used as the trancript_id. Objects that belong to multipe transcripts or genes are duplicated.

This method requires ID and Parent fields to be present.

Because this method reads the whole file in, it uses the most memory, although see –read-twice and –by-chrom for tricks that might help.

set-field

The gene_id and transcript_id fields are set to the value of a provided field. Records that don't have these fields are discarded. By default:

transcript_id=ID gene_id=Parent

set-pattern

As above, but the fieldnames are set by a string format involving the fields of the record.

set-none

transcript_id and gene_id are set to None.

Command line options

```
usage: gff32gtf [-h] [-m {hierarchy,set-field,set-pattern,set-none}]
                [-g GENE_TYPE] [-t TRANSCRIPT_TYPE] [-d]
                [--gene-id GENE_FIELD_OR_PATTERN]
                [--transcript-id TRANSCRIPT_FIELD_OR_PATTERN]
                [--parent-field PARENT] [--read-twice] [--by-chrom]
                [--fail-missing-gene] [--timeit TIMEIT_FILE]
                [--timeit-name TIMEIT_NAME] [--timeit-header]
                [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                [--log-config-filename LOG_CONFIG_FILENAME]
                [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                [-E STDERR] [-S STDOUT]
gff32gtf: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'gff_compare'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'gi2parents'
```

script_template.py

Tags Python

Purpose

Convert the output of a metaphlan analysis to a preferred table format

Usage

Example:

```
python metaphlan2table.py --help
```

Type:

```
python metaphlan2table.py --help
```

for command line help.

Documentation

Code

```
usage: metaphlan2table [-h] [--version] [-t {read_map,rel_ab}]
                       [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                       [--timeit-header] [--random-seed RANDOM_SEED]
                       [-v LOGLEVEL]
                       [--log-config-filename LOG_CONFIG_FILENAME]
                       [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                       [-E STDERR] [-S STDOUT]
metaphlan2table: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'numbers2rgb'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
```

(continues on next page)

(continued from previous page)

```
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'runExpression'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'runMEDIPS'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'runSPP'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'runZinba'
```

split_fasta.py

Tags Python

Purpose

Todo: describe purpose of the script.

Usage

Example:

```
python split_fasta.py --help
```

Type:

```
python split_fasta.py --help
```

for command line help.

Command line options

```
usage: split-fasta [-h] [--version] [-f INPUT_FILENAME] [-i INPUT_PATTERN]
                   [-o OUTPUT_PATTERN] [-n NUM_SEQUENCES] [-m MAP_FILENAME]
                   [-s] [--min-size MIN_SIZE] [--timeit TIMEIT_FILE]
                   [--timeit-name TIMEIT_NAME] [--timeit-header]
                   [--random-seed RANDOM_SEED] [-v LOGLEVEL]
                   [--log-config-filename LOG_CONFIG_FILENAME]
                   [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                   [-E STDERR] [-S STDOUT]
split-fasta: error: argument -?: expected one argument
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'split_genome'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
```

(continues on next page)

(continued from previous page)

```
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'split_genomic_fasta_file'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'split_links'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'tfbs2enrichment'
```

```
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/bin/
↪cgat", line 11, in <module>
    sys.exit(main())
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/site-packages/cgat/cgat.py", line 128, in main
    (file, pathname, description) = imp.find_module(command, [path, ])
  File "/home/docs/checkouts/readthedocs.org/user_builds/cgat-apps/conda/stable/lib/
↪python3.7/imp.py", line 297, in find_module
    raise ImportError(_ERR_MSG.format(name), name=name)
ImportError: No module named 'timeseries2diffgenes'
```

transfac2transfac.py - filter transfac motif files

Tags Python

Purpose

Filter a transfac motif file.

Usage

Example:

```
python cgat_script_template.py
```

Type:

```
python cgat_script_template.py --help
```

for command line help.

Command line options

```
usage: transfac2transfac [-h] [-f FILTER_PREFIX] [-p FILTER_PATTERN]
                         [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
                         [--timeit-header] [--random-seed RANDOM_SEED]
                         [-v LOGLEVEL]
                         [--log-config-filename LOG_CONFIG_FILENAME]
                         [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
                         [-E STDERR] [-S STDOUT]
transfac2transfac: error: argument -?: expected one argument
```

wig2bed.py - convert densities to intervals

Purpose

define intervals based on densities within a bigwig file.

The script currently implements the following methods (`--method`):

threshold output windows that contain values above a certain threshold.

std-above-mean output windows that are a certain number of standard deviations above the mean.

multiple-of-mean output windows that are a certain times above the mean.

Usage

Bigwig files need to be supplied by the –bigwig-file options.

For example:

```
python wig2bed.py --threshold=10 --method=threshold --genome-file=mm10 --bigwig-
↪file=in.bw > out.bed
```

Command line options

```
usage: wig2bed [-h] [-m {threshold,stddev-above-mean,multiple-of-mean}]
               [-g GENOME_FILE] [-t THRESHOLD] [-i bigwig]
               [--timeit TIMEIT_FILE] [--timeit-name TIMEIT_NAME]
               [--timeit-header] [--random-seed RANDOM_SEED] [-v LOGLEVEL]
               [--log-config-filename LOG_CONFIG_FILENAME]
               [--tracing {function}] [-? ?] [-I STDIN] [-L STDLOG]
               [-E STDERR] [-S STDOUT]
wig2bed: error: argument -?: expected one argument
```

7.2.2 Modules

This section documents the modules used in CGAT scripts.

CGAT generic toolboxes

These are the modules that every script or module should use.

Genomics

File formats

Modules for parsing and working for data in specific formats.

AGP.py - working with AGP files

This module contains a parser for reading from *agp* formatted files.

Code

class AGP.**AGP**
Bases: object

Parser for AGP formatted files.

readFromFile(*infile*)
read an agp file.

Example line:

```
scaffold_1      1       1199    1       W       contig_13       1       1199 ␣
↪  +
```

This method converts coordinates to zero-based coordinates using open/closed notation.

In AGP nomenclature (http://www.ncbi.nlm.nih.gov/genome/guide/Assembly/AGP_Specification.html) objects (obj) like scaffolds are assembled from components (com) like contigs.

Component types are:

W WGS sequence

N gap of specified length.

mapLocation(*id*, *start*, *end*)
map a genomic location.

Raises **KeyError** – If *id* is not present.

Bed.py - Tools for working with bed files

This module contains methods for working with *bed* formatted files.

Note: Another way to access the information in *bed* formatted files is through pysam.

The principal class is *Bed* to represent *bed* formatted entries. The method `iterate()` iterates over a bed file and is aware of UCSC track information that might be embedded in the file. Additional functions can process intervals (*merge()*, *binIntervals()*, *setName()*, etc).

The method *readAndIndex()* can build an in-memory index of a bed-file for quick cross-referencing.

Reference

class Bed.**Bed**
Bases: object

an interval in bed format.

Coordinates are represented as 0-based, half-open intervals.

Fields in the record can be accessed as attributes or through a dictionary type access:

```
print b.contig()
print b["contig"]
```

Bed-formatted records can have a variable number of columuns with a minimum of 3. Accessing an optional attribute that is not present will raise an IndexError.

contig
Chromosome/contig.

Type string

start
Start position of the interval.

Type int

end
End position of the interval.

Type int

name
Name of the interval (optional).

Type string

score
Score associated with interval (optional).

Type float

strand
Strand of the interval (optional).

Type char

thickStart

thickEnd

itemRGB

blockCount
Number of blocks for bed intervals spanning multiple blocks (BED12).

Type int

blockSizes
Comma-separated list of sizes of the blocks (BED12).

Type string

blockStarts
Comma-separated list of start positions of the blocks (BED12).

Type string

copy()
Returns a new bed object that is a copy of this one

fromGTF(*gff*, *is_gtf=False*, *name=None*)
fill fields from gtf formatted entry

Parameters

- **gff** (*a gff entry.*) – The object should contain the fields `contig`, `start` and `end` in 0-based, half-open coordinates.
- **name** (*bool*) – If given, attempt to set the name atttribute of the interval by this attribute of the *gff* object such as `gene_id` or `transcript_id`.

toIntervals()
return intervals for BED12 entries.

If the entry is not BED12, the whole region will be returned.

Returns intervals – A list of tuples (start,end) with the block coordinates in the Bed entry.

Return type list

fromIntervals(*intervals*)
Fill co-ordinates from list of intervals.

If multiple intervals are provided and entry is BED12 then the blocks are automatically set.

Parameters **intervals** (*list*) – List of tuples (start, end) with block coordinates.

property columns
return number of columns in bed-entry.

class Bed.**Track**(*line*)
Bases: object

Bed track information.

Bed.**iterator**(*infile*)
iterate over a *bed* formatted file.

Comments and empty lines are ignored. The iterator is *track* aware and will set the `track` attribute for the Bed objects it yields.

Parameters **infile** (*File*) –

Yields *bed* – *Bed* object

Bed.**bed_iterator**(*infile*)
Deprecated, use *iterator()*.

Bed.**setName**(*iterator*)
yield bed entries in which name is set to the record number if unset.

Yields *bed* – *Bed* object

Bed.**grouped_iterator**(*iterator*)
yield bed results grouped by track.

Note that the iterator supplied needs to be sorted by the track attribute. This is usually the case in *bed* formatted files.

Yields *bed* – *Bed* object

Bed.**blocked_iterator**(*iterator*)
yield blocked bed results.

Intervals with the same name are merged into a single entry. This method can be used to convert BED6 formatted entries to BED12. Note that the input iterator needs to be sorted by bed name.

Yields *bed* – *Bed* object

Bed.**readAndIndex**(*infile*, *with_values=False*, *per_track=False*)
read and index a bed formatted file in `infile`.

The index is not strand-aware.

Parameters

- **infile** (*File*) – File object to read from.
- **with_values** (*bool*) – If True, store the actual bed entry. Otherwise, just the intervals are recorded and any additional fields will be ignored.
- **per_track** (*bool*) – If True build indices per track.

Returns **index** – A dictionary of nested containment lists (NCL). Each key is a contig. If *per_track* is set, the dictionary has an additional first level for the track.

Return type dict

Bed.**binIntervals**(*iterator*, *num_bins=5*, *method='equal-bases'*, *bin_edges=None*)
merge adjacent intervals by the score attribute.

This method takes all the intervals in the collection builds a histogram of all the scores in the collection. The partition into the bins can use one of the following merging methods:

equal-bases merge intervals such that each bin contains the equal number of bases

equal-intervals merge intervals such that each bin contains the equal number intervals

This method requires the fifth field (score) of the bed input file to be present.

Parameters

- **iterator** – Iterator yielding bed intervals
- **num_bins** (*int*) – Number of bins to create in the histogram
- **method** (*string*) – Binning method
- **bin_edges** (*list*) – List of bin edges. These take precedence over *method*.

Returns

- **intervals** (*list*) – list of intervals (*Bed*)
- **bin_edges** (*list*) – list of bin edges

Bed.**merge**(*iterator*)
merge overlapping intervals and returns a list of merged intervals.

Bed.**getNumColumns**(*filename*)
return number of fields in bed-file by looking at the first entry.

Returns **ncolumns** – The number of columns. If the file is empty, 0 is returned.

Return type int

Blat.py - tools for working with PSL formatted files and data

This module provides a class to parse *PSL* formatted files such as those output by the BLAT tool.

This module defines the *Blat.Match* class representing a single entry and a series of iterators to iterate of *PSL* formatted files (*iterator()*, *iterator_target_overlap()*, ...).

Reference

exception Blat.**Error**
Bases: Exception

Base class for exceptions in this module.

exception Blat.**ParsingError**(*message*, *line=None*)
Bases: *Blat.Error*

Exception raised for errors while parsing

message -- explanation of the error

class Blat.**Match**
Bases: object

a *psl* formatted alignment.

Block coordinates are on the forward strand for target and on the forward/reverse strand for the query depending on the strand.

The fields mQueryFrom/To and mSbjctFrom/To are always on the forward strand.

convertCoordinates()
convert coordinates.

This rescales the block positions so that they start at 0 and converts the query to forward and the sbjct to forward/reverse coordinates.

About the psl psl format from the manual at http://genome.ucsc.edu/google/goldenPath/help/pslSpec.html

:: In general the coordinates in psl files are "zero based half open." The first base in a sequence is numbered zero rather than one. When representing a range the end coordinate is not included in the range. Thus the first 100 bases of a sequence are represented as 0-100, and the second 100 bases are represented as 100-200.

There is a another little unusual feature in the .psl format. It has to do with how coordinates are handled on the negative strand. In the qStart/qEnd fields the coordinates are where it matches from the point of view of the forward strand (even when the match is on the reverse strand). However on the qStarts[] list, the coordinates are reversed.

This class works in forward coordinates for the query and forward/reverse coordinates for the sbjct.

For a negative strand match, the following is done:

- invert mSbjctFrom and mSbjctTo with mSbjctLength
- add block sizes to mQueryStarts and mSbjctStarts
- invert mQueryStarts and mSbjctStarts
- reverse blocksize, mQueryStarts and mSbjctStarts

switchTargetStrand()
switch the target strand.

Use in cases in which a feature has been defined on the negative target strand with reverse coordinates. The result will be the same alignment using forward coordinates on the target.

This method will also update the query strand and coordinates.

fromMaq(*maq*)
build BLAT entry from a MAQ match.

see `Maq.Match`.

getBlocks()
return a list of aligned blocks.

getMapQuery2Target()
return a map between query to target.

If the strand is "-", the coordinates for query are on the negative strand.

getMapTarget2Query()
return a map between target to query.

If the strand is "-", the coordinates for query are on the negative strand.

fromMap(*map_query2target*, *use_strand=None*)
return a map between query to target.

fromPair(*query_start*, *query_size*, *query_strand*, *query_seq*, *target_start*, *target_size*, *target_strand*, *target_seq*)
fill from two aligned sequences.

Note that sequences are case-sensitive.

class Blat.**MatchPSLX**
Bases: *Blat.Match*

fromPSL(*other*, *query_sequence*, *sbjct_sequence*)
fill entry from a psl match.

sequences are on forward strand starting at query_from and sbjct_from, respectively.

Blat.**iterator2**(*infile*)
iterate over the contents of a psl file.

Blat.**iterator**(*infile*)
iterate over the contents of a psl file.

Blat.**iterator_pslx**(*infile*)
iterate over the contents of a pslx file.

Blat.**iterator_target_overlap**(*infile*, *merge_distance*)
iterate over psl formatted infile and return blocks of target overlapping alignments.

Blat.**iterator_query_overlap**(*infile*, *merge_distance*)
iterate over psl formatted infile and return blocks of target overlapping alignments.

Blat.**iterator_test**(*infile*, *report_step=100000*)
only output parseable lines from infile.

Blat.**iterator_per_query**(*iterator_psl*)
iterate over the contents of a psl file per query

Blat.**addAlignments**(*matches*, *shift=0*, *by_query=False*)
building a genome to query alignment for all matches

The genome alignment is shifted by *shift*.

Blat.**getComponents**(*matches*, *max_distance=0*, *min_overlap=0*, *by_query=False*)
return overlapping matches.

max_distance allow reads to be joined if they are # residues apart. Adjacent reads are 1 residue apart, overlapping reads are 0 residues apart

min_overlap require at least # residues to be overlapping

CBioPortal.py - Interface with the Sloan-Kettering cBioPortal webservice

The Sloan Kettering cBioPortal webservice provides access to a database of results of genomics experiments on various cancers. The database is organised into studies, each study contains a number of case lists, where each list contains the ids of a set of patients, and genetic profiles, each of which represents an assay conducted on the patients in the case list as part of the study.

The main class here is the CBioPortal class representing a connection to the cBioPortal Database. Query's are represented as methods of the class. Study ids or names or case lists can be provided to the constructor to the object, via the setDefaultStudy and setDefaultCaseList methods or to the indevidual query methods. Where ever possible the validity of parameters is checked *before* the query is executed.

Whenever a query requires a genetic profile id or a list of such ids, but none are given, the list of all profiles for which the show_in_analysis flag is set will be used.

All of the commands provided in the webservice are implemented here and as far as possible the name, syntax and paramter names of the query are identical to the raw commands to the webservice. These queries are:

- getCancerStudies,
- getCaseLists,
- getProfileData,
- getMutationData,
- getClinicalData,
- getProteinArrayInfo,
- getProteinArrayData,
- getLink,
- getOncoprintHTML.

In addition two new queries are implememented that are not part of the webservice:

- getPercentAltered and
- getTotalAltered

These emulate the function of the website where the percent of cases that show any alteration for the gene and profiles given are returned (getPercentAltered, or the percent of cases that show an alteration in any of the genes (getTotalAltered) is returned.

examples:

```
gene_list = [ "TP53",
"BCL2",
"MYC"  ]
portal = CBioPortal()
portal.setDefaultStudy(study = "prad_mskcc")
portal.setDefaultCaseList(case_set_id = "prad_all_complete")
portal.getPercentAltered(gene_list = gene_list)
```

or more tersely:

```
portal.CBioProtal()
portal.getPercentAltered(study = "prad_mskcc", case_set_id = "prad_all_complete",
                         gene_list = ["TP53","BCL2","MYC"],
                         genetic_profile_id =["prad_mskcc_mrna"])
```

Any warnings returned by the query are stored in CBioPortal.last_warnings.

Query's that would give too long an URL are split into smaller querys and the results combined transparently.

A commandline interface is provided for convenience, syntax:

```
python CBioPortal.py [options] command(s)
```

Reference

class CBioPortal.**CBioPortal**(*url=None*, *study=None*, *study_name=None*, *case_list_id=None*)
Bases: object

connect to the cBioPortal Database.

If no url is specified the default url is used. A list of of valid study ids is retrieved from the database. This both confirms that the datavase is reachable, and provides cached checking for the ids provided. If a study or study name is provided then this is set as the defualt study for this session and the details of the availible profiles and cases is retrieved. 'Study' is the study id. If both study and study_name are specified then the study id is used.

getCancerStudies()
Fetches the list of cancer studies currently in the database.

Returns list of dictionaries with three entries 'cancer_study_id','name' and 'description'. Also caches this data to verify the validity of later calls

getGeneticProfiles(*study=None*, *study_name=None*)
Fetches the valid genetic profiles for a particular study.

study is the study id. If both study and study_name are specified, study is used. If neither study nor study name is specified then the default study is used if set, if not a value error is raised. Returns a list of dictionaries

getCaseLists(*study=None*, *study_name=None*)
Retrieves meta-data regarding all case lists stored about a specific cancer study.

For example, a within a particular study, only some cases may have sequence data, and another subset of cases may have been sequenced and treated with a specific therapeutic protocol. Multiple case lists may be associated with each cancer study, and this method enables you to retrieve meta-data regarding all of these case lists.

Data is returned as a list of dictionaries with the following entries:

- case_list_id: a unique ID used to identify the case list ID in subsequent interface calls. This is a human readable ID. For example, "gbm_all" identifies all cases profiles in the TCGA GBM study.
- case_list_name: short name for the case list.
- case_list_description: short description of the case list.
- cancer_study_id: cancer study ID tied to this genetic profile. Will match the input cancer_study_id.
- case_ids: space delimited list of all case IDs that make up this case list.

getProfileData(*gene_list*, *case_set_id=None*, *genetic_profile_id=None*, *study=None*, *study_name=None*)
Retrieves genomic profile data for one or more genes.

You can specify one gene and many profiles or one profile and many genes. If you specify no genetic profiles then all genetic profiles for the specified or default study are used if the case_set_id is from that study otherwise a ValueError is raised.

Return value depends on the parameters. If you specify a single genetic profile and multiple genes a list of ordered dictionaries with the following entries:

```
gene_id: Entrez Gene ID
common: HUGO Gene Symbol
entries 3 - N: Data for each case
```

If you specify multi genetic profiles and a single gene, a list of ordered dictoraries with the following entries is returned:

```
genetic_profile_id: The Genetic Profile ID.
alteration_type: The Genetic Alteration Type, e.g. MUTATION, MUTATION_
↪EXTENDED, COPY_NUMBER_ALTERATION, or MRNA_EXPRESSION.
gene_id: Entrez Gene ID.
common: HUGO Gene Symbol.
Columns 5 - N: Data for each case.
```

getMutationData(*gene_list*, *genetic_profile_id*, *case_set_id=None*, *study=None*, *study_name=None*)

For data of type EXTENDED_MUTATION, you can request the full set of annotated extended mutation data.

This enables you to, for example, determine which sequencing center sequenced the mutation, the amino acid change that results from the mutation, or gather links to predicted functional consequences of the mutation.

Query Format

case_set_id= [case set ID] (required) genetic_profile_id= [a single genetic profile IDs] (required). gene_list= [one or more genes, specified as HUGO Gene Symbols or

Entrez Gene IDs](required)

Response Format

A list of dictionaries with the following entires

entrez_gene_id: Entrez Gene ID. gene_symbol: HUGO Gene Symbol. case_id: Case ID. sequencing_center: Sequencer Center responsible for identifying

this mutation. For example: broad.mit.edu.

mutation_status: somatic or germline mutation status. all mutations returned will be of type somatic.

mutation_type: mutation type, such as nonsense, missense, or frameshift_ins. validation_status: validation status. Usually valid, invalid, or unknown. amino_acid_change: amino acid change resulting from the mutation.

functional_impact_score: predicted functional impact score, as predicted by: Mutation Assessor.

xvar_link: Link to the Mutation Assessor web site. xvar_link_pdb: Link to the Protein Data Bank (PDB) View within

Mutation Assessor web site.

xvar_link_msa: Link the Multiple Sequence Alignment (MSA) view within the Mutation Assessor web site.

chr: chromosome where mutation occurs. start_position: start position of mutation. end_position: end position of mutation.

If a default study is set then a check will be performed to set if the supplied case id is from the specified study. The study can be over written using the study and study_name parameters

getClinicalData(*case_set_id=None*, *study=None*, *study_name=None*)

Retrieves overall survival, disease free survival and age at diagnosis for specified cases.

Due to patient privacy restrictions, no other clinical data is available.

case_set_id= [case set ID] (required)

A list of dictionaries with the following entries:

case_id: Unique Case Identifier. overall_survival_months: Overall survival, in months. overall_survival_status: Overall survival status, usually

indicated as "LIVING" or "DECEASED".

disease_free_survival_months: Disease free survival, in months. disease_free_survival_status: Disease free survival status,

usually indicated as "DiseaseFree" or "Recurred/Progressed".

age_at_diagnosis: Age at diagnosis.

If a study is specified or a defualt study is set, then the case_set_id will be tested to check if it exists for that study.

getProteinArrayInfo(*protein_array_type=None*, *gene_list=None*, *study=None*, *study_name=None*)

Retrieves information on antibodies used by reverse-phase protein arrays (RPPA) to measure protein/phosphoprotein levels.

cancer_study_id= [cancer study ID] (required) protein_array_type= [protein_level or phosphorylation] gene_list= [one or more genes, specified as HUGO Gene Symbols or Entrez Gene IDs].

A list of dictionaries with the following entires:

ARRAY_ID: The protein array ID. ARRAY_TYPE: The protein array antibody type, i.e. protein_level

or phosphorylation.

GENE: The targeted gene name (HUGO gene symbol). RESIDUE: The targeted resdue(s).

If no study is specified the default study is used. If that is not specified an error is raised.

getProteinArrayData(*protein_array_id=None*, *case_set_id=None*, *array_info=0*, *study=None*, *study_name=None*)

Retrieves protein and/or phosphoprotein levels measured by reverse-phase protein arrays (RPPA).

case_set_id= [case set ID] protein_array_id= [one or more protein array IDs] as list. array_info= [1 or 0]. If 1, antibody information will also be exported.

If the parameter of array_info is not specified or it is not 1, returns a list of dictionaries with the following columns.

ARRAY_ID: The protein array ID. Columns 2 - N: Data for each case.

If the parameter of array_info is 1, you will receive a list of ordered dictionaries with the following entires:

ARRAY_ID: The protein array ID. ARRAY_TYPE: The protein array antibody type, i.e. protein_level or

phosphorylation.

GENE: The targeted gene name (HUGO gene symbol). RESIDUE: The targeted resdue(s). Columns 5 - N: Data for each case.

If the defualt study is set then the case_set_id will be check. The default study can be overidden using the study or study_name parameters.

getLink(*gene_list*, *study=None*, *study_name=None*, *report='full'*)

return a perminant link to the cBioPortal report for the gene_list cancer_study_id=[cancer study ID] gene_list=[a comma separated list of HUGO gene symbols] (required) report=[report to display; can be one of: full (default), oncoprint_html]

getOncoprintHTML(*gene_list*, *study=None*, *study_name=None*)
returns the HTML for the oncoprint report for the specified gene list and study

setDefaultStudy(*study=None*, *study_name=None*)
sets a new study as the default study. Will check that the study id is valid

setDefaultCaseList(*case_set_id*, *study=None*, *study_name=None*)
set the default case list. If study is not specified the default study will be used.

The study will be used to check that the case_set exists.

getPercentAltered(*gene_list*, *study=None*, *study_name=None*, *case_set_id=None*, *genetic_profile_id=None*, *threshold=2*)
Get the percent of cases that have one or more of the specified alterations for each gene

study = [cancer_study_id] The study to use.

study_name = [cancer_study_name] The name of the study to use. If neither this nor study are specified, then the default is used.

case_set_id = [case_set_id] The case list to use. If not specified, the default case list is used.

gene_list = [one or more genes, specified as HUGO Gene Symobls or ENtrez Gene IDs] (require)

genetic_profile_id = [one or more genetic profile IDs] If none specified all genetic profiles for the specified study are used..

threhold = [z_score_threshold] the numeric threshold at which a mrna expression z-score is said to be significant.

A list of dictionaries with the following entries gene_id: The Entrez Gene ID common: The Hugo Gene Symbol altered_in: The percent of cases in which the gene is altered

One implementation note is that a guess must be made as to wether a returned profile value represents a alteration or not. Currently guesses are only made for copy number variation, mrna expression and mutionation

getTotalAltered(*gene_list*, *study=None*, *study_name=None*, *case_set_id=None*, *genetic_profile_id=None*, *threshold=2*)
Calculate the percent of cases in which any one of the specified genes are altered

exception CBioPortal.**CDGSError**(*error*, *request*)
Bases: Exception

exception that handles errors returned by querys in the database

Fastalterator.py - Iteration over fasta files

This module provides a simple iterator of Fasta formatted files. The difference to the biopython iterator is that the iterators in this module skip over comment lines starting with "#".

Note: Another way to access the information in *fasta* formatted files is through pysam.

Reference

class FastaIterator.**FastaRecord**(*title*, *sequence*, *fold=False*)
Bases: object

a *fasta* record.

title
the title of the sequence

Type string

sequence
the sequence

Type string

fold
the number of bases per line when writing out

Type int

class FastaIterator.**FastaIterator**(*f*, **args*, ***kwargs*)
Bases: object

a iterator of *fasta* formatted files.

Yields *FastaRecord*

FastaIterator.**iterate**(*infile*, *comment='#'*, *fold=False*)
iterate over fasta data in infile

Lines before the first fasta record are ignored (starting with >) as well as lines starting with the comment character.

Parameters

- **infile** (*File*) – the input file
- **comment** (*char*) – comment character
- **fold** (*int*) – the number of bases before line split when writing out

Yields *FastaRecord*

FastaIterator.**iterate_together**(**args*)
iterate synchronously over one or more fasta files.

The iteration finishes once any of the files is exhausted.

:param *fasta*-formatted files to be iterated upon:

Yields *tuple* – a tuple of *FastaRecord* corresponding to the current record in each file.

FastaIterator.**count**(*filename*)
count number of sequences in fasta file.

This method uses the `grep` utility to count lines starting with >.

Parameters **filename** (*string*) – The filename

Raises **OSError** – If the file does not exist

Returns The number of sequences in the file.

Return type int

Fastq.py - methods for dealing with fastq files

This module provides an iterator of *fastq* formatted files (`iterate()`). Additional iterators allow guessing of the quality score format (`iterate_guess()`) or converting them (`iterate_convert()`) while iterating through a file.

`guessFormat()` inspects a fastq file to guess the quality score format and `getOffset()` returns the numeric offset for quality score conversion for a particular quality score format.

Note: Another way to access the information in *fastq* formatted files is through pysam.

Reference

class Fastq.**Record**(*identifier*, *seq*, *quals*, *format=None*)

Bases: object

A record representing a *fastq* formatted record.

identifier

Sequence identifier

Type string

seq

Sequence

Type string

quals

String representation of quality scores.

Type string

format

Quality score format. Can be one of `sanger`, `illumina-1.8`, `solexa` or `phred64`.

Type string

guessFormat()

return quality score format - might return several if ambiguous.

guessDataType()

return the datatype. This is done by inspecting the sequence for basecalls/colorspace ints

trim(*trim3*, *trim5=0*)

remove nucleotides/quality scores from the 3' and 5' ends.

trim5(*trim5=0*)

remove nucleotides/quality scores from the 5' ends.

toPhred()

return qualities as a list of phred-scores.

fromPhred(*quals*, *format*)

set qualities from a list of phred-scores.

Fastq.**iterate**(*infile*)

iterate over contents of fastq file.

Fastq.**iterate_guess**(*infile*, *max_tries=10000*, *guess=None*)
iterate over contents of fastq file.

Guess quality format by looking at the first *max_tries* entries and then subsequently setting the quality score format for each entry.

Parameters

- **infile** (*File*) – File or file-like object to iterate over
- **max_tries** (*int*) – Number of records to examine for guessing the quality score format.
- **guess** (*string*) – Default format. This format will be chosen in the quality score format is ambiguous. The method checks if the *guess* is compatible with the records read so far.

Yields *fastq* – An object of type *Record*.

Raises **ValueError** – If the ranges of the fastq records are not compatible, are incompatible with guess or are ambiguous.

Fastq.**iterate_convert**(*infile*, *format*, *max_tries=10000*, *guess=None*)
iterate over contents of fastq file.

The quality score format is guessed and all subsequent records are converted to *format*.

Parameters

- **infile** (*File*) – File or file-like object to iterate over
- **format** (*string*) – Quality score format to convert all records into.
- **max_tries** (*int*) – Number of records to examine for guessing the quality score format.
- **guess** (*string*) – Default format. This format will be chosen in the quality score format is ambiguous. The method checks if the *guess* is compatible with the records read so far.

Yields *fastq* – An object of type *Record*.

Raises **ValueError** – If the ranges of the fastq records are not compatible, are incompatible with guess or are ambiguous.

Fastq.**guessFormat**(*infile*, *max_lines=10000*, *raises=True*)
guess format of FASTQ File.

Parameters

- **infile** (*File*) – File or file-like object to iterate over
- **max_lines** (*int*) – Number of lines to examine for guessing the quality score format.
- **raises** (*bool*) – Raise ValueError if format is ambiguous

Returns **formats** – list of quality score formats compatible with the file

Return type list

Raises **ValueError** – If the ranges of the fastq records are not compatible.

Fastq.**guessDataType**(*infile*, *max_lines=10000*, *raises=True*)
guess datatype of FASTQ File from [colourspace, basecalls]

Parameters

- **infile** (*File*) –
- **or file-like object to iterate over** (*File*) –
- **max_lines** (*int*) – Number of lines to examine for guessing the datatype

- **raises** (*bool*) – Raise ValueError if format is ambiguous

Returns **formats** – list of datatypes compatible with the file (should only ever be one!)

Return type list

Raises **ValueError** – If the ranges of the fastq records are not compatible.

Fastq.**getOffset**(*format*, *raises=True*)
returns the ASCII offset for a certain format.

If *raises* is set a ValueError is raised if there is not a single offset. Otherwise, a minimum offset is returned.

Returns **offset** – The quality score offset

Return type int

Fastq.**getReadLength**(*filename*)
return readlength from a fastq file.

Only the first read is inspected. If there are different read lengths in the file, the result will be inaccurate.

Returns **read_length**

Return type int

GFF3 - Classes, functions and iterators for working with GFF3 files

This module mostly inherits from the *GTF* and replaces selected functionality to allow working with GFF3 formatted files.

class GFF3.**Entry**
Bases: cgat.GTF.Entry

representation of a GFF3 formatted entry.

This class inherits from *GTF.Entry*, but changes the parsing to reflect GFF3.

parseInfo(*attributes*, *line=None*)
Parse the attributes line of an entry, line parameter provided purely for backwards compatability

getAttributeField()
return the attributes field as a ; delimied field

GFF3.**flat_file_iterator**(*infile*)
simple iterator that iterators over lines in a field and yeilds GFF3 Entry objects

GFF3.**iterator_from_gff**(*gff_iterator*)
to make this slot in with other gtf using scripts, allow copying of an entry into gff3 format. Acts via str, probably not the most efficient way to do things

GFF3.**chrom_iterator**(*gff3_iterator*)
takes a an iterator and returns an iterator over iterators, with a new instance every time a new chromosome is found

GTF.py - Classes and methods for dealing with GTF/GFF formatted files

The coordinates are kept internally in python coordinates (0-based, open-closed), but are output as inclusive 1-based coordinates according to http://www.sanger.ac.uk/Software/formats/GFF/.

The default GTF version is 2.2.

This module uses pysam to provide the principal engine for iterating over files (`iterate()`). As a consequence, the returned objects are of type `pysam.GTFProxy()`.

The class defined in this model *Entry* is useful for re-formatting records.

Apart from basic iteration, this module provides the following utilities:

- Additional iterators for grouping/modifying *GTF* formatted files: *track_iterator()*, *chunk_iterator()*, *iterator_contigs()*, *transcript_iterator()*, *joined_iterator()*, *gene_iterator()*, *flat_gene_iterator()*, *merged_gene_iterator()*, *iterator_filtered()*, *iterator_sorted_chunks()*, *iterator_min_feature_length()*, *iterator_sorted()* *iterator_overlapping_genes()*, *iterator_transcripts2genes()* *iterator_overlaps()*
- Compare intervals: *Identity()*, *HalfIdentity()*, *Overlap()*
- Read GTF formatted files and optionally index them: *readFromFile()*, *readAsIntervals()*, *readAndIndex()*
- Manipulate lists of GTF records: *asRanges()*, *CombineOverlaps()*, *SortPerContig()*, *toIntronIntervals()*, *toSequence()*

GTF.**iterator**(*infile*)
: return a simple iterator over all entries in a file.

GTF.**track_iterator**(*infile*)
: a simple iterator over all entries in a file.

GTF.**chunk_iterator**(*gff_iterator*)
: iterate over the contents of a gff file.

  return entries as single element lists

GTF.**iterator_contigs**(*gffs*)
: iterate over contigs.

  TODO: implement as coroutines

GTF.**transcript_iterator**(*gff_iterator*, *strict=True*)
: iterate over the contents of a gtf file.

  return a list of entries with the same transcript id.

  Any features without a transcript_id will be ignored.

  The entries for the same transcript have to be consecutive in the file. If *strict* is set an AssertionError will be raised if that is not true.

GTF.**joined_iterator**(*gff_iterator*, *group_field=None*)
: iterate over the contents of a gff file.

  return a list of entries with the same group id. Note: the entries have to be consecutive in the file.

GTF.**gene_iterator**(*gff_iterator*, *strict=True*)
: iterate over the contents of a gtf file.

  return a list of transcripts with the same gene id.

Note: the entries have to be consecutive in the file, i.e, first sorted by transcript and then by gene id.

Genes with the same name on different contigs are resolved separately in *strict* = False.

GTF.**flat_gene_iterator**(*gff_iterator*, *strict=True*)
iterate over the contents of a gtf file.

return a list of entries with the same gene id.

Note: the entries have to be consecutive in the file, i.e, sorted by gene_id

Genes with the same name on different contigs are resolved separately in *strict* = False

GTF.**merged_gene_iterator**(*gff_iterator*)
iterate over the contents of a gtf file.

Each gene is merged into a single entry spanning the whole stretch that a gene covers.

Note: the entries have to be consecutive in the file, i.e, sorted by gene_id

GTF.**iterator_filtered**(*gff_iterator*, *feature=None*, *source=None*, *contig=None*, *interval=None*, *strand=None*)
iterate over the contents of a gff file.

yield only entries for a given feature

GTF.**iterator_sorted_chunks**(*gff_iterator*, *sort_by='contig-start'*)
iterate over chunks in a sorted order

sort_by can be

contig-start sort by position ignoring the strand

contig-strand-start sort by position taking the strand into account

contig-strand-start-end intervals with the same start position will be sorted by end position

returns the chunks.

GTF.**iterator_min_feature_length**(*gff_iterator*, *min_length*, *feature='exon'*)
select only those genes with a minimum length of a given feature.

GTF.**iterator_sorted**(*gff_iterator*, *sort_order='gene'*)
sort input and yield sorted output.

GTF.**iterator_overlapping_genes**(*gtf_iterator*, *min_overlap=0*)
return overlapping genes.

GTF.**iterator_transcripts2genes**(*gtf_iterator*, *min_overlap=0*)
cluster transcripts by exon overlap.

The gene id is set to the first transcript encountered of a gene. If a gene stretches over several contigs, subsequent copies are appended a number.

GTF.**iterator_overlaps**(*gff_iterator*, *min_overlap=0*)
iterate over gff file and return a list of features that are overlapping.

The input should be sorted by contig,start

GTF.**Overlap**(*entry1*, *entry2*, *min_overlap=0*)
returns true, if entry1 and entry2 overlap by a minimum number of residues.

GTF.**Identity**(*entry1*, *entry2*, *max_slippage=0*)
returns true, if entry1 and entry2 are (almost) identical, allowing a small amount of slippage at either end.

GTF.**HalfIdentity**(*entry1*, *entry2*, *max_slippage=0*)
returns true, if entry1 and entry2 overlap and at least one end is within *max_slippage* residues.

GTF.**asRanges**(*gffs*, *feature=None*)
return ranges within a set of gffs.

Overlapping intervals are merged.

The returned intervals are sorted.

GTF.**CombineOverlaps**(*old_gff*, *method='combine'*)
combine overlapping entries for a list of gffs.

method can be any of combine|longest|shortest only the first letter is important.

GTF.**SortPerContig**(*gff*)
sort gff entries per contig and return a dictionary mapping a contig to the begin of the list.

GTF.**toIntronIntervals**(*chunk*)
convert a set of gtf elements within a transcript to intron coordinates.

Will use first transcript_id found.

Note that coordinates will still be forward strand coordinates

GTF.**toSequence**(*chunk*, *fasta*)
convert a list of gff attributes to a single sequence.

This function ensures correct in-order concatenation on positive/negative strand. Overlapping regions are merged.

GTF.**readFromFile**(*infile*)
read records from file and return as list.

GTF.**readAsIntervals**(*gff_iterator*, *with_values=False*, *with_records=False*, *merge_genes=False*, *with_gene_id=False*, *with_transcript_id=False*, *use_strand=False*)
read tuples of (start, end) from a GTF file.

This method ignores everything else but the coordinates.

The *with_values*, *with_gene_id* and *with_records* options are exclusive.

Parameters

- **gff_iterator** (*iterator*) – Iterator yielding GTF records.
- **with_values** – If True, the content of the score field is added to the tuples.
- **with_records** – If True, the entire record is added to the tuples.
- **merge_genes** – If true, the GTF records are passed through the :func: *merged_gene_iterator* iterator first.
- **with_gene_id** – If True, the gene_id is added to the tuples.
- **with_transcript_id** – If True, the transcript_ids are added to the tuples.
- **use_strand** – If true, intervals will be grouped by contig and strand. The default is to group by contig only.
- **a dictionary of intervals by contig.** (*Returns*) –

GTF.**readAndIndex**(*iterator*, *with_value=True*)
read from gtf stream and index.

Returns an object of type *IndexedGenome.IndexedGenome*

Return type index

exception GTF.**Error**
Bases: Exception

Base class for exceptions in this module.

exception GTF.**ParsingError**(*message*)
Bases: *GTF.Error*

Exception raised for errors in the input.

message -- explanation of the error

GTF.**toDot**(*v*)
convert value to '.' if None

GTF.**quote**(*v*)
return a quoted attribute.

class GTF.**Entry**
Bases: object

representation of a *GTF* formatted entry.

contig
Chromosome/contig

Type string

source
The GTF source field

Type string

feature
The GTF feature field

Type string

frame
The frame

Type string

start
Start coordinate in 0-based coordinates, half-open coordinates

Type int

end
End coordinate in 0-based coordinates, half-open coordinates

Type int

score
Score associated with feature

Type float

strand
Strand of feature

Type string

gene_id
Gene identifier of feature. Not present for *GFF* formatted data.

Type string

transcript_id
Transcript identifier of feature. Not present for *GFF* formatted data.

Type string

attributes
Dictionary of additional attributes in the GFF/GTF record (last column)

Type dict

read(*line*)
read gff entry from line in GTF/GFF format.

<seqname> <source> <feature> <start> <end> <score> <strand> <frame> [attributes] [comments]

parseInfo(*attributes*, *line*)
parse attributes.

This method will set the gene_id and transcript_id attributes if present.

invert(*lcontig*)
invert genomic coordinates from forward to reverse coordinates and back.

Parameters **lcontig** (*int*) – Length of the chromosome that the feature resides on.

fromGTF(*other*, *gene_id=None*, *transcript_id=None*)
fill record from other GFF/GTF entry.

The optional attributes are not copied.

fromBed(*other*, ***kwargs*)
fill record from a bed entry.

copy(*other*)
fill from other entry.

This method works if other is *GTF.Entry* or `pysam.GTFProxy`.

asDict()
return attributes as a dictionary.

hasOverlap(*other*, *min_overlap=0*)
returns true, if overlap with other entry.

isIdentical(*other*, *max_slippage=0*)
returns true, if self and other overlap completely.

isHalfIdentical(*other*, *max_slippage=0*)
returns true, if self and other overlap.

IndexedFasta.py - fast random access in fasta files

This module provides fast random access to *fasta* formatted files that have been previously indexed. The indexing can be done either through the samtools faidx tool (accessible through pysam) or using the in-house methods implemented in this module.

The main class is *IndexedFasta*. This is a factory function that provides transparent access to both samtools or cgat indexed fasta files. The basic usage to retrieve the sequence spanning the region chr12:10,000-10,100 is:

```
from IndexedFasta import IndexedFasta
fasta = IndexedFasta("hg19")
fasta.getSequence("chr12", "+", 10000, 10100)
```

To index a file, use the `scripts/index_fasta` command line utility or the *createDatabase()* function:

```
> python index_fasta.py hg19 chr*.fa
```

This module has some useful utility functions:

splitFasta() split a *fasta* formatted file into smaller pieces.

parseCoordinates() parse a coordinate string in various formats

but otherwise the module contains a multitude of additional functions that are only of internal use.

Reference

IndexedFasta.**writeFragments**(*outfile_fasta*, *outfile_index*, *fragments*, *mangler*, *size*, *write_all=False*)

write mangled fragments to *outfile_fasta* in chunks of *size* updating *outfile_index*.

returns part of last fragment that has not been written and is less than *size* and the number of fragments output.

If *write_all* is True, all of the fragments are written to the file and the last file position is added to *outfile_index* as well.

class IndexedFasta.**Translator**

Bases: object

translate a sequence.

class IndexedFasta.**TranslatorPhred**(**args*, ***kwargs*)

Bases: *IndexedFasta.Translator*

translate phred quality scores.

class IndexedFasta.**TranslatorSolexa**(**args*, ***kwargs*)

Bases: *IndexedFasta.Translator*

translate solexa quality scores.

class IndexedFasta.**TranslatorRange200**(**args*, ***kwargs*)

Bases: *IndexedFasta.Translator*

translate pcap quality scores.

For example for PCAP scores.

These scores range from 0 to 100 and are the "a weighted sum of input base quality values (Huang and Madan 1999)

The numerical values from 0 to 200 are stored as values form 33 to 233 "

class IndexedFasta.**TranslatorBytes**(**args*, ***kwargs*)

Bases: *IndexedFasta.Translator*

output binary values as bytes permitting values from 0 to 255

Note the resulting file will not be iterable as newline is not a record-separator any more.

IndexedFasta.**createDatabase**(*db*, *iterator*, *force=False*, *synonyms=None*, *compression=None*, *random_access_points=None*, *regex_identifier=None*, *clean_sequence=False*, *ignore_duplicates=False*, *allow_duplicates=False*, *translator=None*)

index files in filenames to create database.

Two new files are created - db.fasta and db_name.idx

If compression is enabled, provide random access points every # bytes.

Dictzip is treated as an uncompressed file.

regex_identifier: pattern to extract identifier from description line. If None, the part until the first white-space character is used.

translator: specify a translator

class IndexedFasta.**cgatIndexedFasta**(*dbname*)
Bases: object

an indexed fasta file.

setTranslator(*translator=None*)
set the *Translator* to use.

getDatabaseName()
returns the name of the database.

getToken(*contig*)
check if token is in index.

getLength(*contig*)
return sequence length for sbjct_token.

getLengths()
return all sequence lengths.

compressIndex()
compress index. Creates a database interface to an index.

getContigs()
return a list of contigs (no synonyms).

getContigSizes(*with_synonyms=True*)
return hash with contig sizes including synonyms.

setConverter(*converter*)
set converter from coordinate system to 0-based, both strand, open/closed coordinate system.

getSequence(*contig*, *strand='+'*, *start=0*, *end=0*, *converter=None*, *as_array=False*)
get a genomic fragment.

A genomic fragment is identified by the coordinates contig, strand, start, end.

The converter function supplied translated these coordinates into 0-based coordinates. By default, start and end are assumed to be pythonic coordinates and are forward/reverse coordinates.

If as_array is set to true, return the AString object. This might be beneficial for large sequence chunks. If as_array is set to False, return a python string.

getRandomCoordinates(*size*)
returns coordinates for a random fragment of size #.

The coordinates are forward/reverse.

Default sampling mode:

Each residue has the same probability of being in a fragment. Thus, the fragment can be smaller than size due to contig boundaries.

class IndexedFasta.**PysamIndexedFasta**(*dbname*)
Bases: *IndexedFasta.cgatIndexedFasta*

interface a pysam/samtools indexed fasta file with the cgatIndexedFasta API.

getSequence(*contig*, *strand='+'*, *start=0*, *end=0*, *converter=None*, *as_array=False*)
get a genomic fragment.

A genomic fragment is identified by the coordinates contig, strand, start, end.

The converter function supplied translated these coordinates into 0-based coordinates. By default, start and end are assumed to be pythonic coordinates and are forward/reverse coordinates.

If as_array is set to true, return the AString object. This might be beneficial for large sequence chunks. If as_array is set to False, return a python string.

IndexedFasta.**IndexedFasta**(*dbname*, **args*, ***kwargs*)
factory function for IndexedFasta objects.

IndexedFasta.**getConverter**(*format*)
return a converter function for converting various coordinate schemes into 0-based, both strand, closed-open ranges.

converter functions have the parameters x, y, s, l: with x and y the coordinates of a sequence fragment, s the strand (True is positive) and l being the length of the contig.

Format is a "-" separated combination of the keywords "one", "zero", "forward", "both", "open", "closed":

zero/one: zero or one-based coordinates forward/both: forward coordinates or forward/reverse coordinates open/closed: half-open intervals (pythonic) or closed intervals

IndexedFasta.**benchmarkRandomFragment**(*fasta*, *size*)
returns a random fragment of size.

IndexedFasta.**verify**(*fasta1*, *fasta2*, *num_iterations*, *fragment_size*, *stdout=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*, *quiet=False*)
verify two databases.

Get segment from fasta1 and check for presence in fasta2.

IndexedFasta.**splitFasta**(*infile*, *chunk_size*, *dir='/tmp'*, *pattern=None*)
split a fasta file into a subset of files.

If pattern is not given, random file names are chosen.

IndexedFasta.**parseCoordinates**(*s*)
parse a coordinate string.

The coordinate string can be various formats, such as `chr1:+:10:1000`, `chr1:10..1000`.

Returns

- **contig** (*string*) – The chromosome/contig.
- **strand** (*char*) – Strand. If not present, set to "+".
- **start** (*int*) – Start of interval
- **end** (*int*) – End of interval. If not present, set to start + 1.

IndexedGenome.py - Random access to interval lists

This module provides a consistent front-end to various interval containers.

Two implementations are available:

NCL Nested containment lists as described in http://bioinformatics.oxfordjournals.org/content/23/11/1386.short. The implemenation was taken from pygr.

quicksect Quicksect algorithm used in Galaxy, see here. This requires python.bx to be installed. The benefit of quicksect is that it allows also quick retrieval of intervals that are closest before or after an query.

The principal clas is *IndexedGenome* which uses NCL and stores a value associated with each interval. *Quicksect* is equivalent to *IndexedGenome* but uses quicksect. The *Simple* is a light-weight version of *IndexedGenome* that does not store a value and thus preserves space.

The basic usage is:

```
from IndexedGenome import IndexedGenome
index = IndexedGenome()
for contig, start, end, value in intervals:
   index.add(contig, start, end, value)

print index.contains("chr1", 1000, 2000)
print index.get("chr1", 10000, 20000)
```

The index is built in memory.

Reference

class IndexedGenome.**IndexedGenome**
Bases: object

Genome with indexed intervals.

index_factory
alias of cgat.NCL.NCL

get(*contig*, *start*, *end*)
return intervals overlapping with key.

class IndexedGenome.**Simple**(**args*, ***kwargs*)
Bases: *IndexedGenome.IndexedGenome*

index intervals without storing a value.

index_factory
alias of cgat.NCL.NCLSimple

class IndexedGenome.**Quicksect**(**args*, ***kwargs*)
Bases: *IndexedGenome.IndexedGenome*

index intervals using quicksect.

Permits finding closest interval in case there is no overlap.

get(*contig*, *start*, *end*)
return intervals overlapping with key.

before(*contig*, *start*, *end*, *num_intervals=1*, *max_dist=2500*)
get closest interval before *start*.

after (*contig*, *start*, *end*, *num_intervals=1*, *max_dist=2500*)
get closest interval after *end*.

Sra.py - Methods for dealing with short read archive files

Utility functions for dealing with *SRA* formatted files from the Short Read Archive.

Requirements: * fastq-dump >= 2.1.7

Code

Sra.**peek** (*sra*, *outdir=None*)
return the full file names for all files which will be extracted

Parameters **outdir** (path) – perform extraction in outdir. If outdir is None, the extraction will take place in a temporary directory, which will be deleted afterwards.

Returns

- **files** (*list*) – A list of fastq formatted files that are contained in the archive.
- **format** (*string*) – The quality score format in the *fastq* formatted files.

Sra.**extract** (*sra*, *outdir*, *tool='fastq-dump'*)
return statement for extracting the SRA file in *outdir*. possible tools are fastq-dump and abi-dump. Use abi-dump for colorspace

Sra.**prefetch** (*sra*)
Use prefetch from the SRA toolkit to download the local cache

Sra.**clean_cache** (*sra*)
Remove the specified SRA file from the cache.

Sra.**fetch_ENA** (*dl_path*, *outdir*, *protocol='ascp'*)
Fetch fastq from ENA given accession

Sra.**fetch_ENA_files** (*accession*)
Get the names of the files matching the ENA accession

Sra.**fetch_TCGA_fastq** (*acc*, *filename*, *token=None*, *outdir='.'*)
Get Fastq file from TCGA repository. Because of the nature of the TCGA repository it assumes certain things:

1) That data is paired-end fastq

2) That the files end in _1.fastq or _2.fastq

Sra.**fetch_TCGA_BAM** (*acc*, *token*, *outdir='.'*, *filter_bed=None*)
Get BAM file from TCGA repository based on UUID. Will return statement and path/filename of downloaded file. A bed file may be provided to filter to remove contigs not present in the reference genome

Sra.**process_remote_BAM** (*infile*, *token=None*, *outdir='.'*, *filter_bed=None*)
generate statement from .remote file

VCF.py - Tools for working with VCF files

The parser for VCF files is very simplistic.

Note: Another way to access the information in *vcf* formatted files is through pysam.

The Variant Call Format (*vcf*) is described at http://www.1000genomes.org/wiki/doku.php?id=1000_genomes:analysis:vcf4.0

Reference

class VCF.**VCFEntry**(*data*, *samples*)
Bases: object

A VCF Entry

class VCF.**VCFFile**(*infile*)
Bases: object

A VCF File

Algorithms

AString.py - strings as arrays of characters

This module provides the *AString* class to efficiently represent long, chromosomal nucleotide sequences in memory.

Reference

class AString.**AString**(**args*)
Bases: array.array

implementation of a string as an array.

This class conserves memory as it uses only 1 byte per letter, while python strings use the machine word size for a letter.

It adds a subset of the python string class such as upper() and lower() for convenience. Slicing and printing return strings.

The *AString* can be constructed by any iterable that is accepted by the constructor of array.array.

upper()
return upper case version.

lower()
return lower case version.

Genomics.py - Tools for working with genomic data

Tags Python

Reference

Genomics.**parse_region_string**(*s*)
parse a genomic region string.

Returns tuple of contig, start, end. Missing values are None.

Genomics.**reverse_complement**(*s*)
reverse complement a sequence.

```
>>> complement("ACATACATACTA")
'TAGTATGTATGT'
```

Returns

Return type string

Genomics.**GetHID**(*sequence*)
returns a hash value for a sequence.

The hash value is computed using md5 and converted into printable characters.

```
>>> GetHID("ACATACATACTA")
'trcPGx9VNT36XMlG0XvUBQ'
```

Returns

Return type A hash value

Genomics.**String2Location**(*s*)
convert a string to location information.

```
>>> String2Location("chr1:12:15")
('chr1', '+', 12, 15)
```

Returns

- **contig** (*string*)
- **strand** (*string*)
- **start** (*int*)
- **end** (*int*)

Genomics.**readContigSizes**(*infile*)
read sizes of contigs from file.

Parameters **infile** (*string*) – Filename of *tsv* separated file.

Returns

Return type dict

Genomics.**forceForwardCoordinates**(*start*, *end*, *strand*, *length*)
return forward coordinates.

If strand is negative, the coordinates in a and b will be converted. If they are on the positive strand, they will be returned as is.

Parameters

- **start** (*int*) – Start coordinate
- **end** (*int*) – End coordinate
- **strand** (*string*) – Strand of interval. The values of "-", "0", "-1" indicate a negative strand.
- **length** (*int*) – Length of chromosome.

Genomics.**CountGeneFeatures**(*first_position*, *alignment*, *genomic_sequence=None*, *border_stop_codon=0*, *stop_codons=('TAG', 'TAA', 'TGA')*)
calculate number of genomic features in a peptide to genome alignment.

Note that codons can be split, for example:

```
S 0 2 5 0 2 I 0 17541 3 0 2 S 1 2 5 0 2 I 0 27979 3 0 2 S 1 2
```

Parameters

- **first_position** (*int*) – Start of alignment on genome.
- **alignment** (*string*) – Alignment in CIGAR format, for example from **exonerate_**.
- **genomic_sequence** (*string*) – Genomic sequence for alignment
- **border_stop_codon** (*int*) – Number of codons that are ignored at the edges of match regions. border_stop_codon should be divisible by three.
- **stop_codons** (*list*) – List of stop codons

Returns

- **nintrons** (*int*) – Number of introns
- **nframeshifts** (*int*) – Number of frameshifts in aligment.
- **ngaps** (*int*) – Number of gaps in aligment.
- **nsplit** (*int*) – Number of codons split by introns in alignment.
- **nstopcodons** (*int*) – Number of stop codons in alignment.
- **disruptions** (*list*) – List of disruptions in the prediction. Each disruption is a tuple of ("stop|frameshift", position in protein, position in cds, position on genomic sequence).

Genomics.**Alignment2String**(*alignment*)
convert a tuple alignment to an alignment string.

Genomics.**String2Alignment**(*source*)
convert an alignment string to a tuple alignment.

Genomics.**GetAlignmentLength**(*alignment*)
return Alignment length

Genomics.**Alignment2ExonBoundaries**(*alignment*, *query_from=0*, *sbjct_from=0*, *add_stop_codon=1*)
extract exon coordinates from a peptide2genome alignment.

Parameters

- **aligment** (*list*) – List of tuples of the alignment in CIGAR format.
- **query_from** (*int*) – Start position of alignment on peptide sequence.
- **sbjct_from** (*int*) – Start position of alignment on nucleotide sequence.
- **add_stop_codon** (*int*) – Add final stop codon to exon boundaries.

Returns **exons** – A list of exons. Each exon is a tuple of (query_from, query_pos, frame, sbjct_from, sbjct_pos, ali)

Return type list

Genomics.**RemoveFrameShiftsFromAlignment**(*row_ali*, *col_ali*, *gap_char='-'*)

remove frame shifts in an alignment.

Frameshifts are gaps are 1, 2, 4, or 5 residues long.

```
>>> RemoveFrameShiftsFromAlignment("ABC-EFG", "AB-DEFG")
('ABEFG', 'ABEFG')
```

Parameters

- **row_ali** (*string*) – Alignment string of row.
- **col_ali** (*string*) – Alignment string of column.
- **gap_char** (*string*) – Gap character to identify aligments.

Returns

- **new_row_ali** (*string*) – New alignment string for row
- **new_col_ali** (*string*) – New aligment string for column

Genomics.**MaskStopCodons**(*sequence*, *stop_codons=('TAG', 'TAA', 'TGA')*)

mask stop codons in a sequence.

Stop codons are masked with NNN.

Parameters

- **sequence** (*string*) – Nucleotide sequence to mask.
- **stop_codons** (*string*) – List of known stop codons.

Returns **masked_sequence**

Return type string

Genomics.**Alignment2DNA**(*alignment*, *query_from=0*, *sbjct_from=0*)

convert a peptide2genome alignment to a nucleotide2nucleotide alignment.

Instead of peptide coordinates, the alignment will be in codon coordinates.

Parameters

- **aligment** (*list*) – List of tuples of the alignment in CIGAR format.
- **query_from** (*int*) – Start position of alignment on peptide sequence.
- **sbjct_from** (*int*) – Start position of alignment on nucleotide sequence.

Returns **alignment** – The alignment as an alignlib.AlignmentVector object.

Return type object

Genomics.**encodeGenotype**(*code*)
encode genotypes like GG, GA into a one-letter code. The returned code is lower case if code[0] < code[1], otherwise it is uppercase.

Genomics.**decodeGenotype**(*code*)
decode single letter genotypes like m, M into two letters. This is the reverse operation to *encodeGenotype()*.

Genomics.**resolveAmbiguousNA**(*code*)
resolve ambiguous nucleic acid letters.

Genomics.**resolveReverseAmbiguousNA**(*genotype*)
map a genotype to a single letter amino acid amiguous code, for example, CT -> Y.

Genomics.**GetMapAA2Codons**()
returns a map of amino acids to codons

No stop codons. .

Genomics.**MapCodon2AA**(*codon*, *is_seleno=False*, *ignore_n=True*)
map a codon to an amino acid using the standard translation tables

The mapping returns gaps as gaps and will return an amino acid for incomplete codons if there is unambiguous mapping.

If `is_seleno` is set, the codon is translated for a selenoprotein.

If `ignore_n` is set, codons with `n` are returned as `?` in order to distinguish them from stop codons.

Amino acids are returned as upper-case letters.

Genomics.**Alignment2PeptideAlignment**(*alignment*, *query_from=0*, *sbjct_from=0*, *genomic_sequence=None*)
convert a Peptide2DNA aligment to a Peptide2Peptide alignment.

How to handle frameshifts?

Genomics.**translate**(*sequence*, *is_seleno=False*, *prefer_lowercase=True*, *ignore_n=False*)
convert DNA sequence to a peptide sequence

If `is_seleno` is set, "TGA" codons are treated as encoding for selenocysteine.

If `ignore_n` is set, codons with `n` are returned as `?` in order to distinguish them from stop codons.

Genomics.**TranslateDNA2Protein**(**args*, ***kwargs*)
convert a DNA sequence to a peptide sequence. keep case.

deprecated - use *translate()* instead.

Genomics.**Alignment2CDNA**(*alignment*, *query_from=0*, *sbjct_from=0*, *genome=None*, *remove_frameshifts=0*)
build cDNA sequence from genomic fragment and return alignment of query to it.

Genomics.**Exons2Alignment**(*exons*)
build an cigar alignment string from a list of exons.

Genomics.**AlignmentProtein2CDNA**(*src*, *exons1=None*, *exons2=None*)
convert a peptide alignment to a nucleotide alignment.

multiplies coordinates with 3. Insert introns.

Note: alignment starts at 1

Genomics.**GetDegenerateSites**(*seq1*, *seq2*, *degeneracy=4*, *position=3*)
returns two new sequenes containing only degenerate sites.

Only unmutated positions are counted.

class Genomics.**SequencePairInfo**
Bases: object

the first characters are ACGT.

getGCContent()
return GC content.

class Genomics.**SequencePairInfoCodons**
Bases: *Genomics.SequencePairInfo*

the first characters are ACGT.

Genomics.**AlignedPair2SubstitutionMatrix**(*seq1*, *seq2*, *alphabet*)
given a pair of sequences, calculate a substitution matrix for the given alphabet.

Genomics.**CalculatePairIndices**(*seq1*, *seq2*, *gap_char='-'*, *with_codons=False*)
returns number of idential and transitions/transversions substitutions in the alignment.

If with-codons = True, synonymous and nonsynonymous changes will be recorded as well. The routine assumes no frame-shifts and will count more than one change as non-synonymous.

Genomics.**makeSubstitutionMatrix**(*type='EMBOSS'*)
make alignator with DNA substitution matrix.

EMBOSS style matrix: identity = 5 mismatch = -4 gop = -16 gep = -4

ClustalW style matrix: match = 1 mismatch = 0 gop = -10 gep = -0.1

Genomics.**CalculateRCSUValuesFromCounts**(*counts*, *pseudo_counts=0*)
calculate RCSU values for codons.

RCSU = relative frequency / uniform frequency

Genomics.**CalculateCodonFrequenciesFromCounts**(*counts*, *pseudo_counts=0*)
calculate codon frequencies from codon counts per amino acid. pseudo_counts are added if desired.

Genomics.**CalculateCAIWeightsFromCounts**(*counts*, *pseudo_counts=0*)
calculate CAI weights from codon counts. pseudo_counts are added if desired.

Genomics.**IsJunk**(*contig*)
returns true, if contigs is likely to be junk.

This is done by name matching. Junk contigs contain either one of the following:

random, unknown, chrU, chU.

Genomics.**CountCodons**(*sequence*)
count the codons in a sequence.

Genomics.**GetUniformCodonUsage**()
get list of frequencies for codons expected for uniform codon usage.

Genomics.**GetBiasedCodonUsage**(*bias=1.0*)
get list of frequencies for codons according to some bias.

The first codon for each aa is the most biased, all others are less biased.

The ratio determines the relative bias between the first and all other codons. 0.0 is no bias, 1.0 is complete bias.

Genomics.**convertStrand**(*strand*)
convert various strand notations into [+-.].

Genomics.**GetIntronType**(*sequence*, *both_strands=False*)
return intron type for an intronic sequence.

If both_strands is True, both strands are checked.

Genomics.**printPrettyAlignment**(*seq1*, **args*)
print a pretty alignment.

Genomics.**ReadPeptideSequences**(*infile*, *filter=None*, *as_array=False*, *regex_identifier=None*)
read peptide sequence from fasta infile.

Genomics.**ParseFasta2Hash**(*infile*, *filter=None*, *regex_identifier=None*)
read fasta formatted sequences file and build a hash.

Keys are all characters before the first whitespace in the description line.

Previously, if the key contained a ":", everything before the ":" was removed. This is not true any more.

Use array for higher space efficiency.

If regex_identifier is given, this is used to extract the identifier from the fasta description line.

Intervals.py - Utility functions for working with intervals

This module contains utility functions for working intervals or list of intervals.

An interval is a tuple of a start and end coordinate in python's 0-based, half-open notation such as:

```
(12, 20)
```

An interval list is simply a list of such intervals.

The majority of the functions in this module take one or more lists of intervals and return one or more new lists of intervals.

Reference

Intervals.**getLength**(*intervals*)
return sum of intervals lengths.

```
>>> getLength([(10,20), (30,40)])
20
```

Intervals.**combine**(*intervals*)
combine overlapping and adjacent intervals.

```
>>> combine([(10,20), (30,40)])
[(10, 20), (30, 40)]
>>> combine([(10,20), (20,40)])
[(10, 40)]
>>> combine([(10,20), (15,40)])
[(10, 40)]
```

Intervals.**prune**(*intervals*, *first=None*, *last=None*)
truncates all intervals that are extending beyond first or last.

Empty intervals are deleted.

Intervals.**complement**(*intervals*, *first=None*, *last=None*)
complement a list of intervals with intervals not in list.

```
>>> complement([(10,20), (15,40)])
[]
>>> complement([(10,20), (30,40)])
[(20, 30)]
>>> complement([(10,20), (30,40)], first=5)
[(5, 10), (20, 30)]
```

Parameters

- **intervals** (*list*) – List of intervals
- **first** (*int*) – First position. If given, the interval from *first* to the first position in *intervals* is added.
- **last** (*int*) – Last position. If given, the interval from the last position in *intervals* to *last* is added.

Returns **intervals** – A new list of intervals

Return type list

Intervals.**addComplementIntervals**(*intervals*, *first=None*, *last=None*)
complement a list of intervals with intervals not in list and return both.

The resulting interval list is sorted.

Intervals.**joined_iterator**(*intervals1*, *intervals2*)
iterate over the combination of two intervals.

returns the truncated intervals delineating the ranges of overlap between intervals1 and intervals2.

Intervals.**intersect**(*intervals1*, *intervals2*)
intersect two interval sets.

Return a set of intervals that is spanned by intervals in both sets. Returns the union of the two intervals.

Intervals.**truncate**(*intervals1*, *intervals2*)
truncate intervals in intervals1 by intervals2

Example: truncate([(0,5)], [(0,3)]) = [(3,5)]

Intervals.**calculateOverlap**(*intervals1*, *intervals2*)
calculate overlap between two list of intervals.

The intervals within each set should not be overlapping.

Intervals.**fromArray**(*a*)
get intervals from a binary array.

Intervals.**combineAtDistance**(*intervals*, *min_distance*)
combine a list intervals and merge those that are less than a certain distance apart.

Intervals.**getIntersections**(*intervals*)
return regions were two intervals are overlapping.

Intervals.**RemoveIntervalsContained**(*intervals*)
remove intervals that are fully contained in another

[(10, 100), (20, 50), (70, 120), (130, 200), (10, 50), (140, 210), (150, 200)]

results:

[(10, 100), (70, 120), (130, 200), (140, 210)]

Intervals.**RemoveIntervalsSpanning**(*intervals*)
remove intervals that are full covering another, i.e. always keep the smallest.

[(10, 100), (20, 50), (70, 120), (40,80), (130, 200), (10, 50), (140, 210), (150, 200)]

result:

[(20, 50), (40, 80), (70, 120), (150, 200)]

Intervals.**ShortenIntervalsOverlap**(*intervals*, *to_remove*)
shorten intervals, so that there is no overlap with another set of intervals.

assumption: intervals are not overlapping

Motifs.py -

Tags Python

Code

Motifs.**countMotifs**(*infile*, *motifs*)
find regular expression *motifs* in sequences within fasta formatted *infile*.

Motifs.**getCounts**(*matches*)
count numbers of motifs.

Motifs.**getOccurances**(*matches*)
count numbers of motifs, but only once per sequence

Motifs.**iupac2regex**(*pattern*)
convert iupac to regex pattern

Motifs.**regex2iupac**(*pattern*)
convert regex to iupac pattern

SequencePairProperties.py - Computing metrics for aligned sequences

This module provides methods for extracting and reporting sequence properties of aligned nucleotide sequences such as percent identity, substitution rate, etc. Usage is the same as *SequencePairProperties*.

Reference

class SequencePairProperties.**SequencePairPropertiesDistance**(**args*, ***kwargs*)
Bases: SequencePairProperties.SequencePairProperties

base class for distance estimators.

class SequencePairProperties.**SequencePairPropertiesBaseML**(*options*, **args*, ***kwargs*)
Bases: *SequencePairProperties.SequencePairPropertiesDistance*

Counts for nucleic acid sequences.

The first characters are ACGT.

loadPair(*seq1*, *seq2*)
load sequence properties from a pair.

class SequencePairProperties.**SequencePairPropertiesCountsNa**(**args*, ***kwargs*)
Bases: SequencePairProperties.SequencePairProperties

Counts for nucleic acid sequences.

The first characters are ACGT.

buildSubstitutionMatrix(*seq1*, *seq2*, *alphabet*)
given a pair of sequences, calculate a substitution matrix for the given alphabet.

loadPair(*seq1*, *seq2*)
load sequence properties from a pair.

class SequencePairProperties.**SequencePairPropertiesCountsCodons**
Bases: *SequencePairProperties.SequencePairPropertiesCountsNa*

the first characters are ACGT.

class SequencePairProperties.**SequencePairPropertiesPID**(**args*, ***kwargs*)
Bases: *SequencePairProperties.SequencePairPropertiesDistance*

Percent identity.

The percent identity is the ratio of the number of identical residues divided by the number of aligned residues.

loadPair(*seq1*, *seq2*)
load sequence properties from a pair.

SequenceProperties.py - Computing metrics of nucleotide sequences

This module provides methods for extracting and reporting sequence properties of nucleotide sequences such as the composition, length, etc.

The classes provide the algorithms to provide the property. They will store the latest result for output. Thus, processing is a two-step procedure:

```
from SequenceProperties import SequencePropertiesLength
from SequenceProperties import SequencePropertiesNA

counters = [SequencePropertiesLength(), SequencePropertiesNA()]

# output column headers
headers = sum(c.getHeaders() for c in counters]
print "      ".join(headers)

for sequence in sequences:
   # load sequence in each counter
   for c in counters:
       c.loadSequence(sequence)
   # output results
   print "   ".join(map(str, counters))
```

This design is useful to compute multiple properties while iterating only once over an input file and output a single, multi-column table.

Note: While useful and in working order, the design of the classes is cumbersome.

Reference

class SequenceProperties.**SequenceProperties**
Bases: object

Base class.

This class is the base class for SequenceProperty objects. Derived classes need to overload most of its methods.

addProperties(*other*)
add properties.

loadSequence(*sequence*, *seqtype='na'*)
load sequence properties from a sequence.

class SequenceProperties.**SequencePropertiesSequence**
Bases: *SequenceProperties.SequenceProperties*

Add properties: the actual sequence.

sequence The sequence

This class outputs the actual sequence supplied.

addProperties(*other*)
add properties.

loadSequence(*sequence*, *seqtype='na'*)
load sequence properties from a sequence.

class SequenceProperties.**SequencePropertiesHid**
Bases: *SequenceProperties.SequenceProperties*

Add properties: a hash of sequence

hid Hash identifier of a sequence

The hash is computed using the md5 algorithm and the resulting byte sequence is then translated into printable characters.

loadSequence(*sequence*, *seqtype='na'*)
load hid sequence properties from a sequence.

class SequenceProperties.**SequencePropertiesLength**
Bases: *SequenceProperties.SequenceProperties*

Add properties: sequence length and number of codons

length Sequence length

ncodons Length in codons

The number of codons is 0 for an amino-acid sequence.

addProperties(*other*)
add properties.

loadSequence(*sequence*, *seqtype='na'*)
load sequence properties from a sequence.

class SequenceProperties.**SequencePropertiesNA**(*reference_usage=[]*)
Bases: *SequenceProperties.SequenceProperties*

Add properties: nucleotide composition

nUnk Number of unknown residues

nA, nC, nG, nT, nGC, nAT Nucleotide counts

pA, pC, pG, pT, pGC, pAT Nucleotide frequencies

addProperties(*other*)
add properties.

loadSequence(*sequence*, *seqtype='na'*)
load sequence properties from a sequence.

class SequenceProperties.**SequencePropertiesDN**(*reference_usage=[]*)
Bases: *SequenceProperties.SequenceProperties*

Add Properties : dinucleotide counts

nAA, nAC, ... Dinucleotide counts

mCountsOthers Unknown dinucleotides

addProperties(*other*)
add properties.

loadSequence(*sequence*, *seqtype='na'*)
load sequence properties from a sequence.

class SequenceProperties.**SequencePropertiesCpg**(*reference_usage=[]*)
Bases: *SequenceProperties.SequencePropertiesNA*, *SequenceProperties.SequencePropertiesDN*

Add Properties : CpG density and observed / expected.

CpG_count Number of CpG in sequence

CpG_density CpG density, number of CpG divided by 2 * sequence length

CpG_ObsExp Ratio of observed to expected number of CpG. The latter is calculated as the product of nC * nG. The ratio is normalized by the sequence length. Set to 0 if no `C` or `G` in sequence.

addProperties(*other*)
add properties.

loadSequence(*sequence*, *seqtype='na'*)
load sequence properties from a sequence.

class SequenceProperties.**SequencePropertiesGaps**(*gap_chars='xXnN'*, **args*, ***kwargs*)
Bases: *SequenceProperties.SequenceProperties*

Add Properties : number of gaps in a sequence

Gaps are identified by unknown characters (`[XN]`)

ngaps Number of gap characters in sequnce

nseq_regions Number of ungapped regions

ngap_regions Number of gapped regions

loadSequence(*sequence*, *seqtype='na'*)
load sequence properties from a sequence.

addProperties(*other*)
add properties.

class SequenceProperties.**SequencePropertiesDegeneracy**
Bases: *SequenceProperties.SequencePropertiesLength*

Add properties : codon degeneracy

nstops Number of stop codons

nsites1d Number of non-degenerate sites

nsites2d, nsites3d, nsites4d Number 2-fold, 3-fold, 4-fold degenerate sites

ngc Number of positions containing either G or C

ngc3 Number of 3rd codon position containing G or C

ngc3 Number of non-degenerate 3rd codon position containing G or C

n2gc3, n3gc3, n4gc3 Number of 2-fold, 3-fold, 4-fold degenerate 3rd codon positions containing G or C

pgc Percentage of positions containing either G or C

pgc3 Percentage of 3rd codon position containing G or C

pgc3 Percentage of non-degenerate 3rd codon position containing G or C

p2gc3, p3gc3, p4gc3 Percentage of 2-fold, 3-fold, 4-fold degenerate 3rd codon positions containing G or C

The degeneracies for amino acids are:

```
2: MW are non-degenerate.
9: EDKNQHCYF are 2-fold degenerate.
1: I is 3-fold degenerate
5: VGATP are 4-fold degenerate.
3: RLS are 2-fold and four-fold degenerate.
   Depending on the first two codons, the codons are counted
   as two or four-fold degenerate codons. This is encoded
   in the file Genomics.py.
```

The number of degenerate sites is computed across all codon positions.

addProperties(*other*)
add properties.

loadSequence(*sequence*, *seqtype='na'*)
load sequence properties from a sequence.

updateProperties()
update fields from counts.

class SequenceProperties.**SequencePropertiesAA**(*reference_usage=[]*)
Bases: *SequenceProperties.SequenceProperties*

Add Properties : amino acid composition of nucleotide sequence.

The codons in the nucleotide sequence are translated into amino acids before counting. The nucleotide sequence must be a multiple of 3.

nA, nC, nD, ... Amino acid counts.

pA, pC, pD, ... Amino acid frequencies.

addProperties(*other*)
add properties.

loadSequence(*sequence*, *seqtype='na'*)
load sequence properties from a sequence.

getHeaders()
Return list of data headers

class SequenceProperties.**SequencePropertiesAminoAcids**(*reference_usage=[]*)
Bases: *SequenceProperties.SequenceProperties*

Add Properties : amino acid composition

nA, nC, nD, ... Amino acid counts.

pA, pC, pD, ... Amino acid frequencies.

addProperties(*other*)
add properties.

loadSequence(*sequence*, *seqtype='na'*)
load sequence properties from a sequence.

class SequenceProperties.**SequencePropertiesCodons**
Bases: *SequenceProperties.SequencePropertiesLength*

Add Properties : codon frequencies

nAAA, nAAC, ... Codon counts

pAAA, pAAC, ... Codon frequencies

addProperties(*other*)
add properties.

loadSequence(*sequence*, *seqtype='na'*)
load sequence properties from a sequence.

class SequenceProperties.**SequencePropertiesCodonUsage**
Bases: *SequenceProperties.SequencePropertiesCodons*

Add properties : Codon usage

The codon frequency is the ratio of the number of times a particular codon is used for a particular amino acid, didived the number of times that particular amino acid appears in the sequence. A ratio of 1.0 means that this particular codon is always used to encode its amino acid, while a frequency of 0.5 means it is used 50% of the times.

rAAA, rAAC, ... Codon frequencies.

addProperties(*other*)
add properties.

class SequenceProperties.**SequencePropertiesCodonTranslator**
Bases: *SequenceProperties.SequencePropertiesCodonUsage*

Add properties : codon sequence is translated into frequencies.

tsequence comma separated list of codon frequencies. The frequencies are in percentages.

addProperties(*other*)
add properties.

loadSequence(*sequence*, *seqtype='na'*)
load sequence properties from a sequence.

class SequenceProperties.**SequencePropertiesBias**(*reference_usage=[]*, *pseudocounts=0*)
Bases: *SequenceProperties.SequencePropertiesCodons*

Add properties : bias measures of codon sequence.

This class outputs metrics showing how biased the codon usage in a particular sequence is compared to a reference codon usage. The reference codon usage is given as a dictionary of codon frequencies and multiple dictionaries can be given to compute the bias against multiple codon usages.

entropy Entropy of the sequence.

ml0, ml1, ... Message length of sequence compared to reference codon usages.

relml0, relml1, ... Relative message length of sequence compared to reference codon usages. The relative message length is the message lenght divided by the number of codons.

relentropy0, relentropy1, ... Relative entropy of sequence compared to reference codon usages. Also called conditional entropy or encoding cost.

kl0, kl1, ... Kullback-Leibler Divergence (relative entropy) of sequence compared to reference codon usages.

Parameters

- **reference_usage** (*list*) – A list of codon frequency tables. The bias will be computed against each.
- **pseudocounts** (*int*) – Pseudo-counts to add

getMessageLength(*usage*)
return message length of a sequence in terms of its reference usage.

getEntropy(*usage=None*)
return entropy of a source in terms of a reference usage. Also called conditional entropy or encoding cost.

Note that here I compute the sum over 20 entropies, one for each amino acid.

If not given, calculate entropy.

getKL(*usage*)
return Kullback-Leibler Divergence (relative entropy) of sequences with respect to reference codon usage.

class SequenceProperties.**SequencePropertiesCounts**(*alphabet*)
Bases: *SequenceProperties.SequenceProperties*

Add Properties : Residue counts against arbirtrary alphabet

nUnk Number of unknown residues

nA, nB, ... Character counts

pA, pB, ... Character frequencies

Parameters **alphabet** (*string*) – List of characters in alphabet

addProperties(*other*)
add properties.

loadSequence(*sequence*, *seqtype='na'*)
load sequence properties from a sequence.

class SequenceProperties.**SequencePropertiesEntropy**(*alphabet*, *pseudocounts=0*)
Bases: *SequenceProperties.SequencePropertiesCounts*

Add properties : Entropy of a sequence

entropy Entropy of the sequence

Parameters

- **alphabet** (*string*) – List of characters in alphabet
- **pseudocounts** (*int*) – Pseudo-counts to add

addProperties(*other*)
add properties.

getEntropy(*usage=None*)
return entropy of a source in terms of a reference usage.

Also called conditional entropy or encoding cost.

Variants.py -

Tags Python

Code

class Variants.**Variant**(*pos*, *reference*, *genotype*)
Bases: tuple

Create new instance of Variant(pos, reference, genotype)

property genotype
Alias for field number 2

property pos
Alias for field number 0

property reference
Alias for field number 1

Variants.**ExtendedVariant**
alias of *Variants.Variant*

Variants.**updateVariants**(*variants*, *lcontig*, *strand*, *phased=True*)
update variants such that they use same coordinate system (and strand) as the transcript

fixes 1-ness of variants

Variants.**mergeVariants**(*variants*)
merge overlapping variants.

Overlapping variants occur if there are two deletions at the same location:

WT ACTG Allele1 -CT- Allele2 —-

This will be encoded by samtools as (0-based coordinates):

```
0 * -A/ACTG
3 * -G/-G
```

This upsets the re-constitution algoritm.

This method separates these two variants into two non-overlapping variants making use of variable length deletions.

0 * -A/-A 1 * —G/-CTG

Another case:

WT ACTG Allele1 ACT- Allele2 —-

This will be encoded by samtools as (0-based coordinates):

```
0 * */-ACTG
3 * -G/*
```

This method separates these two as:

```
0 * */-ACT
3 * -G/-G
```

`Variants.`**`indexVariants`**(*variants*)
build index of variants for ranged retrieval.

`Variants.`**`buildAlleles`**(*sequence*, *variants*, *reference_start=0*, *phased=True*)
build alleles for `sequence` adding `variants`.

Variants are assumed to be in 0-based coordinates on the same strand as the sequence. `reference_start` is the position of the first base of `sequence`. Set to 0, if the positions in `variants` are relative to `sequence`.

`Variants.`**`buildOffsets`**(*variants*, *phased=True*, *contig=None*)
collect coordinate offsets.

This methods takes a set of variants and computes coordinates offsets based on indels.

Conflicting variants will be removed.

Returns a list of variants, a list of removed variants and a list of offsets.

Wrappers

These modules wrap tools and provide routines for parsing their output.

WrapperCodeML.py -

Tags Python

Code

exception `WrapperCodeML.`**`Error`**
Bases: `Exception`

Base class for exceptions in this module.

exception `WrapperCodeML.`**`ParsingError`**(*message*, *line=None*)
Bases: *`WrapperCodeML.Error`*

Exception raised for errors while parsing

message -- explanation of the error

exception `WrapperCodeML.`**`UsageError`**(*message*)
Bases: *`WrapperCodeML.Error`*

Exception raised for errors while starting

message -- explanation of the error

class WrapperCodeML.**CodeMLBranchInfo**(*branch1*, *branch2*, *kaks*, *ka*, *ks*, *ndn*, *sds*, *n*, *s*)
Bases: object

result with branch information.

class WrapperCodeML.**BaseMLResult**
Bases: WrapperCodeML.CodeMLResult

result object for BaseML.

class WrapperCodeML.**CodeMLResultSites**(*num_sequences*, *model*)
Bases: WrapperCodeML.CodeMLResult

result with site specific information.

class WrapperCodeML.**CodeMLResultPairs**
Bases: WrapperCodeML.CodeMLResult

results for a pairwise codeml run.

fromResult(*result*)
build pairwise results from tree.

class WrapperCodeML.**CodeMLResultPair**
Bases: WrapperCodeML.CodeMLResult

results for a pairwise comparison.

class WrapperCodeML.**CodeMLAncestralSequence**(*sequence*, *accuracy_per_site*, *accuracy_per_sequence*)
Bases: object

an ancestral sequence.

class WrapperCodeML.**CodeML**
Bases: object

GetOptions()
return options in pretty format

AddOptions(*parser*)
add options to an OptionParser object.

SetOptions(*options*)
set options from the command line.

WriteAlignment(*mali*)
write alignment in Phylip format.

WriteTree(*tree*)
write tree to file. The root of the tree is removed.

writeControlFile(*outfile*, *filename_sequences='input'*, *filename_output='output'*, *filename_tree=None*, *options={}*)
write a codeml.ctl file into outfile.

parseRst(*inlines*, *result*)
parse lines from rst file.

checkSection(*lines*, *section_start*)
check if section starts with string section_start.

getSection(*lines*, **args*)
check if section starts with string section_start.

parseLog(*lines_log*, *result*)
parse log output.

parseOutput(*lines*, *lines_log=None*, *rst_lines=None*)
parse CodeML output. This is rather tricky, as paml output is as freeformat as it can get. Also, there is a log file and an output file. Proceed sequentially through file.

class WrapperCodeML.**CodeMLSites**
Bases: *WrapperCodeML.CodeML*

parseOutput(*lines*, *lines_log=None*, *rst_lines=None*)
parse codeml output for site-specific analysis.

parseGrids(*lines*, *result*)
parse grid information.

parseSites(*lines*, *result*)
parse site specific model results.

class WrapperCodeML.**CodeMLPairwise**
Bases: *WrapperCodeML.CodeML*

parseLog(*lines_log*, *result*)
parse log output.

This routine collects the rho values for each pair.

parseOutput(*lines*, *lines_log=None*, *rst_lines=None*)
parse codeml output for pairwise rate calculation.

parsePairs(*lines*, *result*)
parse pairwise results.

class WrapperCodeML.**BaseML**
Bases: *WrapperCodeML.CodeML*

AddOptions(*parser*)
add options to an OptionParser object.

SetOptions(*options*)
set options from the command line.

parseOutput(*lines*, *lines_log=None*, *rst_lines=None*)
parse BASEML output. This is rather tricky, as paml output is as freeformat as it can get. Also, there is a log file and an output file. Proceed sequentially through file.

parseFrequencies(*inlines*, *result*)
parse frequency section.

class WrapperCodeML.**Evolver**
Bases: object

interface class for running evolver.

writeControlFile(*outfile*)
write control file to outfile.

fromMali(*mali*)
compute codon table from a multiple alignment.

setUniformFrequencies()
use uniform codon frequencies.

calculateScale(*ds*)
calculate tree scale for a given dS value.

The branch scale is given by:

t = 3 dS * ps + 3 omega * dS * (1-ps) t = 3 dS * (ps + omega (1 - ps)

setTree(*tree*)
set tree.

run(*ds=None*, *tree=None*, *test=False*, *dump=False*)
run evolver.

class WrapperCodeML.**EvolverBaseml**(**args*, ***kwargs*)
Bases: *WrapperCodeML.Evolver*

interface class for running evolver for nucleotides.

setUniformFrequencies()
use uniform codon frequencies.

fromMali(*mali*)
compute frequencies from a multiple alignment.

getParameters()
get parameters for a model.

From the MCbase.dat: Parameter kappa or rate parameters in the substituton model: For TN93, two kappa values are required, while for REV, 5 values (a,b,c,d,e) are required (see Yang 1994 for the definition of these parameters). The kappa parameter is defined differently under HKY85 (when k=1 means no transition bias) and under F84 (when k=0 means no bias). JC69 and F81 are considered species cases of HKY85, so use 1 for kappa for those two models. Notation is from my two papers in JME in 1994.

writeControlFile(*outfile*)
write control file to outfile.

WrapperCodeML.**getOptions**(*options*)
translate command line options to PAML options.

WrapperCodeML.**runEvolver**(*options*)
run evolver.

IGV.py - Simple wrapper to the IGV socket interface

Tags Python

This code was written by Brent Pedersen.

Downloaded from https://github.com/brentp/bio-playground/blob/master/igv/igv.py on Nov.30 2011.

IGV.**startIGV**(*command='igv.sh'*, *port=None*)
start IGV on a specific port.

class IGV.**IGV**(*host='127.0.0.1'*, *port=60151*, *snapshot_dir='/tmp/igv'*)
Bases: object

Simple wrapper to the IGV (http://www.broadinstitute.org/software/igv/home) socket interface (http://www.broadinstitute.org/software/igv/PortCommands)

requires:

1) **you have IGV running on your machine (launch with webstart here:** http://www.broadinstitute.org/software/igv/download)

2) **you have enabled port communication in** View -> Preferences... -> Advanced

Successful commands return 'OK'

example usage:

```
>>> igv = IGV()
>>> igv.genome('hg19')
'OK'
```

```
>>> igv.load('http://www.broadinstitute.org/igvdata/1KG/pilot2Bams/NA12878.SLX.bam
↪')
'OK'
>>> igv.go('chr1:45,600-45,800')
'OK'
```

#save as svg, png, or jpg

```
>>> igv.save('/tmp/r/region.svg')
'OK'
>>> igv.save('/tmp/r/region.png')
'OK'
```

 # go to a gene name.

```
>>> igv.go('muc5b')
'OK'
>>> igv.sort()
'OK'
>>> igv.save('muc5b.png')
'OK'
```

 # get a list of commands that will work as an IGV batch script.

```
>>> print "\n".join(igv.commands)
snapshotDirectory /tmp/igv
genome hg19
goto chr1:45,600-45,800
snapshotDirectory /tmp/r
snapshot region.svg
snapshot region.png
goto muc5b
sort base
snapshot muc5b.png
```

Note, there will be some delay as the browser has to load the annotations at each step.

sort(*option='base'*)
options is one of: base, position, strand, quality, sample, and readGroup.

Masker.py - Wrapper for sequence masking tools

Tags Python

Code

class Masker.**Masker**
Bases: object

a masker preserves gaps, but it does not preserve whitespace characters.

getAlphabet(*sequence*)
get sequence type (aa,na,codons).

maskSequence(*peptide_sequence*)
mask peptide sequence

maskSequences(*sequences*)
mask a collection of sequences.

class Masker.**MaskerBias**
Bases: *Masker.Masker*

class Masker.**MaskerSeg**
Bases: *Masker.Masker*

class Masker.**MaskerDustMasker**
Bases: *Masker.Masker*

use dustmasker. masked chars are returned as lower case characters.

class Masker.**MaskerRandom**(*proportion=10*, **args*, ***kwargs*)
Bases: *Masker.Masker*

randomly mask a proportion of positions in a sequence in multiple alignment.

Masker.**maskSequences**(*sequences*, *masker=None*)
return a list of masked sequence.

***masker* can be one of** dust/dustmasker * run dustmasker on sequences softmask * use softmask to hardmask sequences

Data processing

Math and Stats

Histogram.py - Various functions to deal with histograms

Author

Tags Python

Histograms can be calculated from a list/tuple/array of values. The histogram returned is then a list of tuples of the format [(bin1,value1), (bin2,value2), . . .].

Histogram.**CalculateFromTable**(*dbhandle*, *field_name*, *from_statement*, *num_bins=None*, *min_value=None*, *max_value=None*, *intervals=None*, *increment=None*)
get a histogram using an SQL-statement. Intervals can be either supplied directly or are build from the data by providing the number of bins and optionally a minimum or maximum value.

If no number of bins are provided, the bin-size is 1.

This command uses the INTERVAL command from MYSQL, i.e. a bin value determines the upper boundary of a bin.

Histogram.**CalculateConst**(*values*, *num_bins=None*, *min_value=None*, *max_value=None*, *intervals=None*, *increment=None*, *combine=None*)
calculate a histogram based on a list or tuple of values.

Histogram.**Calculate**(*values*, *num_bins=None*, *min_value=None*, *max_value=None*, *intervals=None*, *increment=None*, *combine=None*, *no_empty_bins=0*, *dynamic_bins=False*, *ignore_out_of_range=True*)
calculate a histogram based on a list or tuple of values.

use scipy for calculation.

Histogram.**Scale**(*h*, *scale=1.0*)
rescale bins in histogram.

Histogram.**convert**(*h*, *i*, *no_empty_bins=0*)
add bins to histogram.

Histogram.**Combine**(*source_histograms*, *missing_value=0*)
combine a list of histograms Each histogram is a sorted list of bins and counts. The counts can be tuples.

Histogram.**Print**(*h*, *intervalls=None*, *format=0*, *nonull=None*, *format_value=None*, *format_bin=None*)
print a histogram.

A histogram can either be a list/tuple of values or a list/tuple of lists/tuples where the first value contains the bin and second contains the values (which can again be a list/tuple).

format 0 = print histogram in several lines 1 = print histogram on single line

Histogram.**Write**(*outfile*, *h*, *intervalls=None*, *format=0*, *nonull=None*, *format_value=None*, *format_bin=None*)
print a histogram.

A histogram can either be a list/tuple of values or a list/tuple of lists/tuples where the first value contains the bin and second contains the values (which can again be a list/tuple).

Parameters **format** – output format. 0 = print histogram in several lines, 1 = print histogram on single line

Histogram.**Fill**(*h*)
fill every empty value in histogram with previous value.

Histogram.**Add**(*h1*, *h2*)
adds values of histogram h1 and h2 and returns a new histogram

Histogram.**SmoothWrap**(*histogram*, *window_size*)
smooth histogram by sliding window-method, where the window is wrapped around the borders. The sum of all values is entered at center of window.

Histogram.**PrintAscii**(*histogram*, *step_size=1*)
print histogram ascii-style.

Histogram.**Count**(*data*)
count categorized data. Returns a list of tuples with (count, token).

Histogram.**Accumulate**(*h*, *num_bins=2*, *direction=1*)
add successive counts in histogram. Bins are labelled by group average.

Histogram.**Cumulate**(*h*, *direction=1*)
calculate cumulative distribution.

Histogram.**AddRelativeAndCumulativeDistributions**(*h*)
adds relative and cumulative percents to a histogram.

Histogram.**histogram**(*values*, *mode=0*, *bin_function=None*)
Return a list of (value, count) pairs, summarizing the input values. Sorted by increasing value, or if mode=1, by decreasing count. If bin_function is given, map it over values first. Ex: vals = [100, 110, 160, 200, 160, 110, 200, 200, 220] histogram(vals) ==> [(100, 1), (110, 2), (160, 2), (200, 3), (220, 1)] histogram(vals, 1) ==> [(200, 3), (160, 2), (110, 2), (100, 1), (220, 1)] histogram(vals, 1, lambda v: round(v, -2)) ==> [(200.0, 6), (100.0, 3)]

Histogram.**cumulate**(*histogram*)
cumulate histogram in place.

histogram is list of (bin, value) or (bin, (values,))

Histogram.**normalize**(*histogram*)
normalize histogram in place.

histogram is list of (bin, value) or (bin, (values,))

Histogram.**fill**(*iterator*, *bins*)
fill a histogram from bins.

The values are given by an iterator so that the histogram can be built on the fly.

Description:

Count the number of times values from array a fall into numerical ranges defined by bins. Range x is given by bins[x] <= range_x < bins[x+1] where x =0,N and N is the length of the bins array. The last range is given by bins[N] <= range_N < infinity. Values less than bins[0] are not included in the histogram.

Parameters

- **-- The iterator.** (*iterator*) –
- **-- 1D array. Defines the ranges of values to use during** (*bins*) –
- **histogramming.** –

Returns: 1D array. Each value represents the occurences for a given bin (range) of values.

Histogram.**fillHistograms**(*infile*, *columns*, *bins*)
fill several histograms from several columns in a file.

The histograms are built on the fly.

Description:

Count the number of times values from array a fall into numerical ranges defined by bins. Range x is given by bins[x] <= range_x < bins[x+1] where x =0,N and N is the length of the bins array. The last range is given by bins[N] <= range_N < infinity. Values less than bins[0] are not included in the histogram.

Parameters

- **-- The input file.** (*file*) –
- **-- columns to use** (*columns*) –

- **-- a list of 1D arrays. Defines the ranges of values to use during** (*bins*) –
- **histogramming.** –

Returns: a list of 1D arrays. Each value represents the occurences for a given bin (range) of values.

WARNING: missing value in columns are ignored

Histogram2D.py - functions for handling two-dimensional histograms.

Tags Python

Histogram2D.**Calculate**(*values*, *mode=0*, *bin_function=None*)
Return a list of (value, count) pairs, summarizing the input values. Sorted by increasing value, or if mode=1, by decreasing count.

If bin_function is given, map it over values first.

Histogram2D.**Print**(*h*, *bin_function=None*)
print a histogram.

A histogram can either be a list/tuple of values or a list/tuple of lists/tuples where the first value contains the bin and second contains the values (which can again be a list/tuple).

Parameters **format** – output format. 0 = print histogram in several lines, 1 = print histogram on single line

Stats.py - statistical utility functions

Tags Python

Code

Stats.**getSignificance**(*pvalue*, *thresholds=[0.05, 0.01, 0.001]*)
return cartoon of significance of a p-Value.

class Stats.**Result**
Bases: object

allow both member and dictionary access.

Stats.**doLogLikelihoodTest**(*complex_ll*, *complex_np*, *simple_ll*, *simple_np*, *significance_threshold=0.05*)
perform log-likelihood test between model1 and model2.

Stats.**doBinomialTest**(*p*, *sample_size*, *observed*, *significance_threshold=0.05*)
perform a binomial test.

Given are p: the probability of the NULL hypothesis, the sample_size and the number of observed counts.

Stats.**doChiSquaredTest**(*matrix*, *significance_threshold=0.05*)
perform chi-squared test on a matrix.

The observed/expected values are in rows, the categories are in columns, for example:

set	protein_coding	intronic	intergenic
observed	92	90	194
expected	91	10	15

If there are only two categories (one degrees of freedom) the Yates correction is applied. For each entry (observed-expected), the value 0.5 is subtracted ignoring the sign of the difference.

The test throws an exception if

1. one or more expected categories are less than 1 (it does not matter what the observed values are)
2. more than one-fifth of expected categories are less than 5

Stats.**doPearsonChiSquaredTest**(*p*, *sample_size*, *observed*, *significance_threshold=0.05*)
perform a pearson chi squared test.

Given are p: the probability of the NULL hypothesis, the sample_size and the number of observed counts.

For large sample sizes, this test is a continuous approximation to the binomial test.

class Stats.**DistributionalParameters**(*values=None*, *format='%6.4f'*, *mode='float'*)
Bases: object

a collection of distributional parameters. Available properties are:

mMean, mMedian, mMin, mMax, mSampleStd, mSum, mCounts

This method is deprecated - use *Summary* instead.

updateProperties(*values*)
update properties.

If values is an vector of strings, each entry will be converted to float. Entries that can not be converted are ignored.

getZScore(*value*)
return zscore for value.

setFormat(*format*)
set number format.

getHeaders()
returns header of column separated values.

getHeader()
returns header of column separated values.

class Stats.**Summary**(*values=None*, *format='%6.4f'*, *mode='float'*, *allow_empty=True*)
Bases: *Stats.Result*

a collection of distributional parameters. Available properties are:

mean, median, min, max, samplestd, sum, counts

getHeaders()
returns header of column separated values.

getHeader()
returns header of column separated values.

Stats.**doFDRPython**(*pvalues*, *vlambda=None*, *pi0_method='smoother'*, *fdr_level=None*, *robust=False*, *smooth_df=3*, *smooth_log_pi0=False*, *pi0=None*, *plot=False*)
modeled after code taken from http://genomics.princeton.edu/storeylab/qvalue/linux.html.

I did not like the error handling so I translated most to python.

Compute FDR after method by Storey et al. (2002).

class Stats.**CorrelationTest**(*s_result=None*, *method=None*)
Bases: object

coefficient is r, not r squared

Stats.**filterMasked**(*xvals*, *yvals*, *missing=('na', 'Nan', None, '')*, *dtype=<class 'float'>*)
convert xvals and yvals to numpy array skipping pairs with one or more missing values.

Stats.**doCorrelationTest**(*xvals*, *yvals*)
compute correlation between x and y.

Raises a value-error if there are not enough observations.

Stats.**getPooledVariance**(*data*)
return pooled variance from a list of tuples (sample_size, variance).

Stats.**computeROC**(*values*)
return a roc curve for *values*. Values is a sorted list of (value, bool) pairs.

Deprecated - use getPerformance instead

returns a list of (FPR,TPR) tuples.

class Stats.**PairedTTest**(*statistic*, *pvalue*)
Bases: tuple

Create new instance of PairedTTest(statistic, pvalue)

property pvalue
Alias for field number 1

property statistic
Alias for field number 0

Stats.**doPairedTTest**(*vals1*, *vals2*)
perform paired t-test.

vals1 and vals2 need to contain the same number of elements.

Stats.**doWelchsTTest**(*n1*, *mean1*, *std1*, *n2*, *mean2*, *std2*, *alpha=0.05*)
Welch"s approximate t-test for the difference of two means of heteroscedasctic populations.

This functions does a two-tailed test.

see PMID: 12016052

Parameters

n1 [int] number of variates in sample 1

n2 [int] number of variates in sample 2

mean1 [float] mean of sample 1

mean2 [float] mean of sample 2

std1 [float] standard deviation of sample 1

std2 [float] standard deviation of sample 2

returns a WelchTTest

Stats.**getAreaUnderCurve**(*xvalues*, *yvalues*)
compute area under curve from a set of discrete x,y coordinates using trapezoids.

This is only as accurate as the density of points.

Stats.**getSensitivityRecall**(*values*)
return sensitivity/selectivity.

Values is a sorted list of (value, bool) pairs.

Deprecated - use getPerformance instead

class Stats.**ROCResult**(*value*, *pred*, *tp*, *fp*, *tn*, *fn*, *tpr*, *fpr*, *tnr*, *fnr*, *rtpr*, *rfnr*)
Bases: tuple

Create new instance of ROCResult(value, pred, tp, fp, tn, fn, tpr, fpr, tnr, fnr, rtpr, rfnr)

property fn
Alias for field number 5

property fnr
Alias for field number 9

property fp
Alias for field number 3

property fpr
Alias for field number 7

property pred
Alias for field number 1

property rfnr
Alias for field number 11

property rtpr
Alias for field number 10

property tn
Alias for field number 4

property tnr
Alias for field number 8

property tp
Alias for field number 2

property tpr
Alias for field number 6

property value
Alias for field number 0

Stats.**getPerformance**(*values*, *skip_redundant=True*, *false_negatives=False*, *bin_by_value=True*, *monotonous=False*, *multiple=False*, *increasing=True*, *total_positives=None*, *total_false_negatives=None*)
compute performance estimates for a list of `(score, flag)` tuples in *values*.

Values is a sorted list of (value, bool) pairs.

If the option *false-negative* is set, the input is +/- or 1/0 for a true positive or false negative, respectively.

TP: true positives FP: false positives TPR: true positive rate = true_positives / predicted P: predicted FPR: false positive rate = false positives / predicted value: value

Stats.**doMannWhitneyUTest**(*xvals*, *yvals*)
apply the Mann-Whitney U test to test for the difference of medians.

Stats.**adjustPValues**(*pvalues*, *method='fdr'*, *n=None*)
returns an array of adjusted pvalues

Reimplementation of p.adjust in the R package.

p: numeric vector of p-values (possibly with 'NA's). Any other R is coerced by 'as.numeric'.

method: correction method. Valid values are:

n: number of comparisons, must be at least 'length(p)'; only set this (to non-default) when you know what you are doing

For more information, see the documentation of the p.adjust method in R.

Stats.**savitzky_golay**(*y*, *window_size*, *order*, *deriv=0*, *rate=1*)
Smooth (and optionally differentiate) data with a Savitzky-Golay filter. The Savitzky-Golay filter removes high frequency noise from data. It has the advantage of preserving the original shape and features of the signal better than other types of filtering approaches, such as moving averages techniques.

Parameters

- **y** (*array_like, shape (N,)*) – the values of the time history of the signal.
- **window_size** (*int*) – the length of the window. Must be an odd integer number.
- **order** (*int*) – the order of the polynomial used in the filtering. Must be less then *window_size* - 1.
- **deriv** (*int*) – the order of the derivative to compute (default = 0 means only smoothing)

Returns **ys** – the smoothed signal (or it's n-th derivative).

Return type ndarray, shape (N)

Notes

The Savitzky-Golay is a type of low-pass filter, particularly suited for smoothing noisy data. The main idea behind this approach is to make for each point a least-square fit with a polynomial of high order over a odd-sized window centered at the point.

Examples

t = np.linspace(-4, 4, 500) y = np.exp(-t**2) + np.random.normal(0, 0.05, t.shape) ysg = savitzky_golay(y, window_size=31, order=4) import matplotlib.pyplot as plt plt.plot(t, y, label='Noisy signal') plt.plot(t, np.exp(-t**2), 'k', lw=1.5, label='Original signal') plt.plot(t, ysg, 'r', label='Filtered signal') plt.legend() plt.show()

References

MatrixTools.py -

Tags Python

Code

MatrixTools.**addOptions**(*parser*)
add matrices to option parser.

MatrixTools.**getMatrixFromEdges**(*lines*, *options*, *in_map_token2row={}*, *in_map_token2col={}*)
read matrix from lines

MatrixTools.**buildMatrixFromLists**(*lists*, *dtype=<class 'float'>*, *default=None*)
build a matrix from a list of lists.

Each list is a list of tuples (row, value). The columns are given by order of the lists.

Returns matrix, row_headers

MatrixTools.**buildMatrixFromTables**(*infiles*, *column*, *column_header=0*, *dtype=<class 'float'>*, *default=None*)

build a matrix from a column called *column* in a series of input files.

If column_value is None, the first column is taken as the name of the row.

The columns are given by order of the input files.

returns matrix, row_headers

MatrixTools.**buildMatrixFromEdges**(*edges*, *in_map_token2row={}*, *in_map_token2col={}*, *is_symmetric=False*, *missing_value=0*, *diagonal_value=0*, *dtype=<class 'int'>*)

build a matrix from an edge-list representation.

For example, the following list of tuples:

```
[('A', 'B', 1),
 ('A', 'C', 2),
 ('B', 'C', 3)]
```

will be converted to the following matrix:

```
  A B C
A   1 2
B     3
C
```

If *is_symmetric* is set to True, the matrix is assumed to be symmetric and missing values will automatically be filled:

```
  A B C
A   1 2
B 1   3
C 2 3
```

If edge list may contain four elements, in which case the fourth element is expected to be the value of the lower diagonal in a symmetric matrix:

```
[('A', 'B', 1, 4),
 ('A', 'C', 2, 5),
 ('B', 'C', 3, 6)]
```

will yield:

```
  A B C
A   1 2
B 4   3
C 5 6
```

returns a numpy matrix and lists of row and column names.

Toolboxes

Toolboxes for generic problems.

Iterators.py - Iterator functions

A collection of general purpose iterators.

`Iterators.`**`sample`**(*iterable*, *sample_size=None*)

sample # copies from iterator without replacement.

Stores a temporary copy of the items in iterable. The function has thus a possibly high memory footprint and long pre-processing time to yield the first element.

If sample_size is not given, the iterator returns elements in random order (see random.shuffle())

Note: This snippet was downloaded from an unknown source.

`Iterators.`**`group_by_distance`**(*iterable*, *distance=1*)

group integers into non-overlapping intervals that are at most *distance* apart.

```
>>> list( group_by_distance( (1,1,2,4,5,7) ) )
[(1, 3), (4, 6), (7, 8)]
```

```
>>> list( group_by_distance( [] ) )
[]
```

```
>>> list( group_by_distance( [3] ) )
[(3, 4)]
```

```
>>> list( group_by_distance( [3,2] ) )
Traceback (most recent call last):
...
ValueError: iterable is not sorted: 2 < 3
```

Note: This snippet was downloaded from an unknown source.

SetTools.py - Tools for working on sets

Some of the functions in this module precede the `set` datatype in python.

Reference

SetTools.**combinations**(*list_of_sets*)
create all combinations of a list of sets

```
>>> combinations([set((1,2)), set((2,3))])
[((0,), set([1, 2]), set([1, 2])), ((1,), set([2, 3]), set([2, 3]))]
>>> combinations([set((1,2)), set((2,3)), set((3,4))])
[((0,), set([1, 2]), set([1, 2])), ((1,), set([2, 3]), set([2, 3])), ((2,),␣
↪set([3, 4]), set([3, 4])), ((0, 1), set([1, 2, 3]), set([2])), ((0, 2), set([1,␣
↪2, 3, 4]), set([])), ((1, 2), set([2, 3, 4]), set([3]))]
```

Returns **result** – The resut is a list of tuples containing (set_composition, union, intersection)

Return type list

SetTools.**writeSets**(*outfile*, *list_of_sets*, *labels=None*)
output a list of sets as a tab-separated file.

This method build a list of all items contained across all sets and outputs a matrix of 0's and 1's denoting set membership. The items are in the table rows and the sets are in the table columns.

Parameters

- **outfile** (*File*) – File to write to
- **list_of_sets** (*list*) – The list of sets to output
- **labels** (*list*) – List of labels(column names)

SetTools.**unionIntersectionMatrix**(*list_of_sets*)
build union and intersection matrix of a list of sets.

```
>>> unionIntersectionMatrix([set((1,2)), set((2,3))])
array([[0, 1],
       [3, 0]])
>>> unionIntersectionMatrix([set((1,2)), set((2,3)), set((3,4))])
array([[0, 1, 0],
       [3, 0, 1],
       [4, 3, 0]])
```

Parameters **list_of_sets** (*list*) – The list of sets to work with.

Returns **matrix** – The matrix is a list of lists. The upper diagonal of the matrix contains the size of the union of two sets and the lower diagonal the intersection of two sets.

Return type numpy.matrix

SetTools.**getAllCombinations**(**sets*)
generate all combination of elements from a collection of sets.

This method is derived from a python recipe by Zoran Isailovski: http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/410685

```
>>> getAllCombinations(set((1,2)), set((2,3)), set((3,4)))
[(1, 2, 3), (1, 2, 4), (1, 3, 3), (1, 3, 4), (2, 2, 3), (2, 2, 4), (2, 3, 3), (2,␣
↪3, 4)]
```

SetTools.**xuniqueCombinations**(*items*, *n*)
Return a list of unique combinations of items in list.

```
>>> list(xuniqueCombinations([1, 2, 3], 1))
[[1], [2], [3]]
>>> list(xuniqueCombinations([1, 2, 3], 2))
[[1, 2], [1, 3], [2, 3]]
>>> list(xuniqueCombinations([1, 2, 3], 3))
[[1, 2, 3]]
```

SetTools.**compareLists**(*list1*, *list2*)
returns the union and the disjoint members of two lists.

Note: Deprecated Use python sets instead.

Returns

- **unique1** (*set*) – Elements unique in set1
- **unique2** (*set*) – Elements unique in set2
- **common** (*set*) – Elements in both lists.

Tree.py - A phylogenetic tree

The *Tree* is derived from the class from Bio.Nexus.Trees.Tree adding some additional functionality.

Reference

Tree.**updateNexus**(*nexus*)
change trees in a nexus object (see **Biopython_**) to *Tree*.

Tree.**Nop**(*x*)
empty function for tree traversal

class Tree.**Tree**(**args*, ***kwargs*)
Bases: Bio.Nexus.Trees.Tree

A phylogenetic tree.

This class represents a tree using a chain of nodes with on predecessor (=ancestor) and multiple successors (=subclades).

Ntree(self,tree).

root_at_node(*node*, *distance=0*)
root tree at node.

Parameters

- **node** – New root
- **distance** (*float*) – Distance of node to new root.
- **is a subset of the code taken from root_with_outgroup.** (*This*) –

to_string(*support_as_branchlengths=False*, *branchlengths_only=False*, *plain=True*, *write_all_taxa=False*, *branchlength_format='%1.5f'*, *support_format='%1.2f'*, *format='nexus'*)
Return a paup compatible tree line.

Parameters

- **support_as_branchlengths** (*bool*) – If true, output bootstrap support value as branch lengths.
- **branchlengths_only** (*bool*) – Only output branchlengths, no support values
- **plain** (*bool*) – Output plain tree (no branch lengths/support values).
- **write_all_taxa** (*bool*) – If true, internal node names are output
- **branchlength_format** (*string*) – Format to use for branch lengths
- **support_format** (*string*) – Format to use for bootstrap support values
- **format** (*string*) – Either `nexus` on `NHX`.

Returns **tree** – A PAUP compatible tree line.

Return type string

get_nodes(*node_id*)
Return a list of nodes downwards from a node (self, node_id).

The list includes the given node_id.

get_leaves(*node_id*)
Return a list of leaf nodes downward from a node (self, node_id).

root_midpoint()
perform midpoint rooting of tree.

The root is placed at equal distance to the two leaves furthest apart in the tree (centroid of the tree).

getNumLeaves()
return list with number of leaves beyond each node

root_balanced()
perform balanced rooting of tree.

The root is placed such that the number of leaves on either side of the tree is equal (or at most different by 1).

dfs(*node_id*, *pre_function=<function Nop>*, *descend_condition=<function Nop>*, *post_function=<function Nop>*)
dfs tree tree traversal starting at node_id.

Apply functions pre_function at first and post_function at last visit of a node.

writeToFile(*outfile*, *with_branchlengths=True*, *format='nh'*)
write a tree to a file.

truncate(*node_id*, *taxon=None*, *keep_node=None*)
truncate tree at node_id.

This function will not change any branch lengths. If keep is given, single child nodes will be collapsed until keep_node is reached.

relabel(*map_old2new*, *warn=False*)
relabel taxa in tree using the provided mapping.

rescaleBranchLengths(*value*)
rescale branch length so that they sum up to value.

TreeTools.py - Tools for working with trees

This module contains functions to work with gene and/or species trees.

Reference

TreeTools.**Newick2Nexus**(*infile*)
convert newick formatted tree(s) into a nexus object.

Multiple trees are separated by a semicolon. Tree names can be given by fasta-style separators, i.e., lines starting with '>'.

If the token [&&NHX is found in the tree, it is assumed to be output from njtree and support values are added. Support values are added in the format taxon:support:branchlength

Parameters **infile** (*object*) – Input data. Can be a file, a list of lines or a single line.

Returns **nexus**

Return type Bio.Nexus.Nexus

TreeTools.**Nexus2Newick**(*nexus*, *with_branchlengths=True*, *with_names=False*, *write_all_taxa=False*)
convert nexus tree format to newick format.

Parameters

- **nexus** (*Bio.Nexus.Nexus*) – The trees to output
- **with_branch_lengths** (*bool*) – If True, output branchlengths.
- **with_names** (*bool*) – If True, add node names.
- **write_all_taxa** (*bool*) – Ouput taxa for internal nodes.

Returns **output** – Trees in Newick format.

Return type string

TreeTools.**Tree2Newick**(*tree*, *with_branch_lengths=True*, *write_all_taxa=False*)
convert tree to newick format.

TreeTools.**Newick2Tree**(*txt*)
convert tree to nexus format.

TreeTools.**WriteNexus**(*nexus*, ***kwargs*)
write trees in nexus file format.

TreeTools.**GetTaxa**(*tree*)
retrieve all taxa of leaves in a tree.

TreeTools.**GetTaxonomicNames**(*tree*)
get list of taxa.

TreeTools.**MapTaxa**(*tree*, *map_old2new*, *remove_unknown=False*)
update taxa in tree to new taxa.

Parameters

- **tree** (*Tree*) – The tree to update.

- **map_old2new** (*dict*) – Dictionary mapping old taxa to new taxa.
- **remove_unknown** (*bool*) – If true, taxa not in *map_old2new* will be removed.

TreeTools.**Branchlength2Support**(*tree*)
copy values stored as branchlength to into support

The branchlength property is not changed.

This step is necessary when support has been stored as branchlength (e.g. paup), and has thus been read in as branchlength.

TreeTools.**Species2Genes**(*nexus*, *map_species2genes*)
convert a species tree to a gene tree.

Parameters

- **nexus** (*Bio.Nexus.Nexus*) – The trees to work on
- **map_species2genes** (*dict*) – Dictionary mapping species names to gene names

TreeTools.**Genes2Species**(*nexus*, *map_gene2species*)
convert a gene tree into a species tree.

Parameters

- **nexus** (*Bio.Nexus.Nexus*) – The trees to work on
- **map_gene2species** (*dict*) – Dictionary mapping gene names to species names

TreeTools.**BuildMapSpecies2Genes**(*genes*, *pattern_species='^([^|]+)[|]'*)
build a map of species to genes

This method assumes that gene names contain the species name and it can be extracted via a regular expression.

Parameters

- **genes** (*list*) – List of genes
- **pattern_species** (*string*) – Regular expression to extract species name from gene name.

Returns

- **map_species2genes** (*dict*) – Mapping between species to one or more genes
- **map_gene2species** (*dict*) – Mapping between a gene to the species

TreeTools.**GetMonophyleticPairs**(*tree*)
build list of monophyletic pairs in tree.

TreeTools.**GetTaxaForSpecies**(*tree*, *species*, *pattern_species='^([^|]+)[|]'*)
get all taxa of a given species.

This method assumes that node labels contain the species name and it can be extracted via a regular expression.

Parameters

- **genes** (*list*) – List of genes
- **pattern_species** (*string*) – Regular expression to extract species name from gene name.

Returns **taxa** – List of taxa from this species.

Return type list

TreeTools.**IsMonophyleticForSpecies**(*tree*, *species*, *pattern_species='^([^|]+)[|]'*)
check if a tree is monophyletic for a species.

This method assumes that node labels contain the species name and it can be extracted via a regular expression.

Parameters

- **tree** (*Tree*) – Tree to analyse
- **species** (*string*) – Species to check
- **pattern_species** (*string*) – Regular expression to extract species name from gene name.

Returns

Return type bool

TreeTools.**IsMonophyleticForTaxa**(*tree*, *taxa*, *support=None*)
check if a tree is monophyletic for a list of taxa.

Parameters

- **tree** (*Tree*) – Tree to analyse
- **taxa** (*list*) – List of taxa
- **support** (*float*) – Minimum bootstrap support

Returns

Return type bool

TreeTools.**GetLeaves**(*tree*, *node*)
Return leaves in tree below node.

TreeTools.**IsSingleSpecies**(*tree*, *node*, *pattern_species='^([^|]+)[|]'*)
True if taxa below node contain the same species.

TreeTools.**Transcript2GeneTree**(*tree*, *map_transcript2gene*, *map_gene2transcripts*)
convert a transcript tree into a gene tree.

supply a map for mapping transcripts to genes.

The procedure for converting a transcript tree into a gene tree:

If there are two genes, and they are monophyletic, no matter how many transcripts, the order is as follows:

1 Merge all nodes into two, one for each gene.

2 The distance between the genes is the minimum distance observed between two transcripts from different genes. Half of this will be set as the branch length from the gene leaves.

If this is not possible for a set of genes, the procedure will fail and not return a gene tree.

TreeTools.**MapTerminalTaxa**(*tree*, *mapping*)
map taxa in leaves in all trees.

TreeTools.**GetCommonAncestor**(*tree*, *taxa*)
retrieve common ancestor for a list of taxa.

Reroot tree. Check if it is monopyletic. If it is, return root, otherwise, return -1.

TreeTools.**TreeDFS**(*tree*, *node_id*, *pre_function=<function Nop>*, *descend_condition=<function Nop>*, *post_function=<function Nop>*)
BFS tree tree traversal starting at node_id.

Apply functions pre_function at first and post_function at last visit of a node.

TreeTools.**GetMaxIndex**(*tree*)
get maximum node number.

TreeTools.**GetBranchLengths**(*tree*)
return an array with minimum and maximum branch length.

TreeTools.**Reroot**(*tree*, *taxa*)
reroot tree with taxa - the list of taxa does not need to be monophyletic.

TreeTools.**GetSubsets**(*tree*, *node=None*, *with_decoration=True*)
return subsets below a certain node including their height (distance from leaves) and branchlength

TreeTools.**CountBranchPoints**(*tree*, *taxa*)
count the number branch points together with their distances for a given list of taxa.

return a list of branch points

TreeTools.**IsCompatible**(*tree1*, *tree2*)
check if two trees are compatible.

note: this will delete support information.

TreeTools.**Tree2Graph**(*tree*)
return tree as a list of edges in a graph.

TreeTools.**Graph2Tree**(*links*, *label_ancestral_nodes=False*)
build tree from list of nodes.

Assumption is that links always point from parent to child.

TreeTools.**GetAllNodes**(*tree*)
return all nodes in the tree.

TreeTools.**GetDistancesBetweenTaxa**(*tree*, *taxa1*, *taxa2*)
get average branchlength between taxa1 and taxa2.

TreeTools.**PruneTerminal**(*tree*, *taxon*)
Prunes a terminal taxon from the tree.

id_of_previous_node = prune(tree,taxon) If taxon is from a bifurcation, the connecting node will be collapsed and its branchlength added to remaining terminal node. This might be no longer a meaningful value.

direct copy of Nexus.Trees.py - don't know why have a separate method, maybe there was a bug in Nexus.Trees.

TreeTools.**GetSubtree**(*tree*, *node_id*)
return a copy of tree from node_id downwards.

TreeTools.**Unroot**(*tree*)
unroot tree.

TreeTools.**GetSize**(*tree*)
return the length of the tree. This is the maximum node_id + 1.

This quantity is useful for tree traversal while updating a container.

TreeTools.**PruneTree**(*tree*, *taxa*, *keep_distance_to_root=False*)
prune tree: keep only those taxa in list.

TreeTools.**GetNodeMap**(*tree1*, *tree2*)
map nodes between tree1 and tree2.

TreeTools.**ReconciliateByRio**(*gene_tree*, *species_tree*, *extract_species*, *extract_gene=None*, *outgroup_species=None*, *min_branch_length=0.0*)
Gene tree G and species tree S

If outgroup_species is given: trees will be cut of as soon as one of the outgroup species is part of a subtree. The corresponding node type will be out-paralog. Out-paralog relationship is cast upwards.

Input trees are rooted and binary.

Output: gene tree with duplication/speciation assigned to each node.

Initialization:

Number nodes in S in pre-order traversal (root = 1), such that child nodes are always larger than parent nodes.

For each external node g of G, set M(g) to the number of the external node in S with the matching species name.

Recursion:

Visit each internal node g of G in post-order traversal, (i.e. from leaves to root):

```
set a = M(g1) # g1 = first child of current node g
set b = M(g2) # g2 = second child of current node g

while a != b:
    if a > b:
          set a = parent of node a in species tree
    else:
          set b = parent of node b in species tree
set M(g) = a

if M(g) == M(g1) or M(g) == M(g2):
    g is duplication
else:
    g is speciation
```

The algorithm returns an array for each node with its type.

If extract_gene is given, the algorithm will label transcription nodes for alternative transcripts (duplications involving the same gene).

The algorithm has been extended to accomodate the following test cases:

Alternative transcripts Alternative transcripts that span genes from other species are permitted, if at most one gene of the other species is involved.

To avoid over-counting of speciation events, the one subtree with the least species is masked.

If the branch length of a node in the gene tree is shorter than min_branch_length, the resultant node is masked, because the topology might be dodgy.

TreeTools.**CountDuplications**(*gene_tree*, *species_tree*, *node_types*, *extract_species*, *extract_gene=None*)

count duplications.

given are gene and species tree and node types (duplication/speciation)

extract_species gives the species for an OTU in the gene tree

Extract_gene gives the gene for an OTU in the gene tree. If not given, all transcripts are counted as unique.

TreeTools.**GetParentNodeWhereTrue**(*node_id*, *tree*, *stop_function*)

walk up in gene tree and stop where stop_function is true.

The walk finishes at the root.

returns tuple of node and distance.

TreeTools.**GetChildNodesWhereTrue**(*node_id*, *tree*, *stop_function*)
walk down in tree and stop where stop_function is true

The walk finishes at the leaves.

returns a list of tuples of nodes and distance.

TreeTools.**GetDistanceToRoot**(*tree*)
return list with distance to root for each node.

TreeTools.**traverseGraph**(*graph*, *start*, *block=[]*)
traverse graph, go not passed nodes in block.

TreeTools.**convertTree2Graph**(*tree*)
convert tree to a graph.

TreeTools.**calculatePatternsFromTree**(*tree*, *sort_order*)
calculate patterns from a tree.

CGAT infrastructure

Below is a list of modules that are involved in maintainig the CGAT infrastructure such as logging, dependency tracking, etc.

Other

RLE.py - a simple run length encoder

Tags Python

Taken from: http://rosettacode.org/wiki/Run-length_encoding#Python

RLE.**encode**(*input_array*)
encode array or string.

return tuples of (count, value).

```
>>> encode(array.array( "i", (10,10,10,10,20,20,20,20) ) )
[(4, 10), (4, 20)]
```

```
>>> encode("aaaaahhhhhhmmmmmmmuiiiiiiiaaaaaa")
[(5, 'a'), (6, 'h'), (7, 'm'), (1, 'u'), (7, 'i'), (6, 'a')]
```

RLE.**decode**(*lst*, *typecode*)
decode to array

```
>>> decode( [(4, 10), (4, 20)], typecode="i" )
array('i', [10, 10, 10, 10, 20, 20, 20, 20])
```

```
>>> decode( [(5, 'a'), (6, 'h'), (7, 'm'), (1, 'u'), (7, 'i'), (6, 'a')],␣
↪typecode="c" )
array('c', 'aaaaahhhhhhmmmmmmmuiiiiiiiaaaaaa')
```

RLE.**compress**(*input_string*, *bytes=1*)
return compressed stream.

SVGdraw.py - generate SVG drawings

Tags Python

This module has been copied from 3rd party resources.

SVGdraw uses an object model drawing and a method toXML to create SVG graphics by using easy to use classes and methods usualy you start by creating a drawing eg

> d=drawing() #then you create a SVG root element s=svg() #then you add some elements eg a circle and add it to the svg root element c=circle() #you can supply attributes by using named arguments. c=circle(fill='red',stroke='blue') #or by updating the attributes attribute: c.attributes['stroke-width']=1 s.addElement(c) #then you add the svg root element to the drawing d.setSVG(s) #and finaly you xmlify the drawing d.toXml()

this results in the svg source of the drawing, which consists of a circle on a white background. Its as easy as that;) This module was created using the SVG specification of www.w3c.org and the O'Reilly (www.oreilly.com) python books as information sources. A svg viewer is available from www.adobe.com

class SVGdraw.**pathdata**(*x=None*, *y=None*)
Bases: object

class used to create a pathdata object which can be used for a path. although most methods are pretty straightforward it might be useful to look at the SVG specification.

closepath()
ends the path

move(*x*, *y*)
move to absolute

relmove(*x*, *y*)
move to relative

line(*x*, *y*)
line to absolute

relline(*x*, *y*)
line to relative

hline(*x*)
horizontal line to absolute

relhline(*x*)
horizontal line to relative

vline(*y*)
verical line to absolute

relvline(*y*)
vertical line to relative

bezier(*x1*, *y1*, *x2*, *y2*, *x*, *y*)
bezier with xy1 and xy2 to xy absolut

relbezier(*x1*, *y1*, *x2*, *y2*, *x*, *y*)
bezier with xy1 and xy2 to xy relative

smbezier(*x2*, *y2*, *x*, *y*)
smooth bezier with xy2 to xy absolut

relsmbezier(*x2*, *y2*, *x*, *y*)
smooth bezier with xy2 to xy relative

qbezier (*x1*, *y1*, *x*, *y*)
quadratic bezier with xy1 to xy absolut

relqbezier (*x1*, *y1*, *x*, *y*)
quadratic bezier with xy1 to xy relative

smqbezier (*x*, *y*)
smooth quadratic bezier to xy absolut

relsmqbezier (*x*, *y*)
smooth quadratic bezier to xy relative

ellarc (*rx*, *ry*, *xrot*, *laf*, *sf*, *x*, *y*)
elliptival arc with rx and ry rotating with xrot using large-arc-flag and sweep-flag to xy absolut

relellarc (*rx*, *ry*, *xrot*, *laf*, *sf*, *x*, *y*)
elliptival arc with rx and ry rotating with xrot using large-arc-flag and sweep-flag to xy relative

class SVGdraw.**SVGelement** (*type*, *attributes*, *elements*, *text*, *namespace*, ***args*)
Bases: object

Creates a arbitrary svg element and is intended to be subclassed not used on its own. This element is the base of every svg element it defines a class which resembles a xml-element. The main advantage of this kind of implementation is that you don't have to create a toXML method for every different graph object. Every element consists of a type, attribute, optional subelements, optional text and an optional namespace. Note the elements==None, if elements = None:self.elements=[] construction. This is done because if you default to elements=[] every object has a reference to the same empty list.

addElement (*SVGelement*)
adds an element to a SVGelement

SVGelement.addElement(SVGelement)

class SVGdraw.**tspan** (*text=None*, ***args*)
Bases: *SVGdraw.SVGelement*

ts=tspan(text='',**args)

a tspan element can be used for applying formatting to a textsection usage: ts=tspan('this text is bold') ts.attributes['font-weight']='bold' st=spannedtext() st.addtspan(ts) t=text(3,5,st)

class SVGdraw.**tref** (*link*, ***args*)
Bases: *SVGdraw.SVGelement*

tr=tref(link='',**args)

a tref element can be used for referencing text by a link to its id. usage: tr=tref('#linktotext') st=spannedtext() st.addtref(tr) t=text(3,5,st)

class SVGdraw.**spannedtext** (*textlist=None*)
Bases: object

st=spannedtext(textlist=[])

a spannedtext can be used for text which consists of text, tspan's and tref's You can use it to add to a text element or path element. Don't add it directly to a svg or a group element. usage:

ts=tspan('this text is bold') ts.attributes['font-weight']='bold' tr=tref('#linktotext') tr.attributes['fill']='red' st=spannedtext() st.addtspan(ts) st.addtref(tr) st.addtext('This text is not bold') t=text(3,5,st)

class SVGdraw.**rect** (*x=None*, *y=None*, *width=None*, *height=None*, *fill=None*, *stroke=None*, *stroke_width=None*, ***args*)
Bases: *SVGdraw.SVGelement*

r=rect(width,height,x,y,fill,stroke,stroke_width,**args)

a rectangle is defined by a width and height and a xy pair

class SVGdraw.**ellipse**(*cx=None*, *cy=None*, *rx=None*, *ry=None*, *fill=None*, *stroke=None*, *stroke_width=None*, ***args*)
Bases: *SVGdraw.SVGelement*

e=ellipse(rx,ry,x,y,fill,stroke,stroke_width,**args)

an ellipse is defined as a center and a x and y radius.

class SVGdraw.**circle**(*cx=None*, *cy=None*, *r=None*, *fill=None*, *stroke=None*, *stroke_width=None*, ***args*)
Bases: *SVGdraw.SVGelement*

c=circle(x,y,radius,fill,stroke,stroke_width,**args)

The circle creates an element using a x, y and radius values eg

class SVGdraw.**point**(*x*, *y*, *fill='black'*, ***args*)
Bases: *SVGdraw.circle*

p=point(x,y,color)

A point is defined as a circle with a size 1 radius. It may be more efficient to use a very small rectangle if you use many points because a circle is difficult to render.

class SVGdraw.**line**(*x1=None*, *y1=None*, *x2=None*, *y2=None*, *stroke=None*, *stroke_width=None*, ***args*)
Bases: *SVGdraw.SVGelement*

l=line(x1,y1,x2,y2,stroke,stroke_width,**args)

A line is defined by a begin x,y pair and an end x,y pair

class SVGdraw.**polyline**(*points*, *fill=None*, *stroke=None*, *stroke_width=None*, ***args*)
Bases: *SVGdraw.SVGelement*

pl=polyline([[x1,y1],[x2,y2],...],fill,stroke,stroke_width,**args)

a polyline is defined by a list of xy pairs

class SVGdraw.**polygon**(*points*, *fill=None*, *stroke=None*, *stroke_width=None*, ***args*)
Bases: *SVGdraw.SVGelement*

pl=polyline([[x1,y1],[x2,y2],...],fill,stroke,stroke_width,**args)

a polygon is defined by a list of xy pairs

class SVGdraw.**path**(*pathdata*, *fill=None*, *stroke=None*, *stroke_width=None*, *id=None*, ***args*)
Bases: *SVGdraw.SVGelement*

p=path(path,fill,stroke,stroke_width,**args)

a path is defined by a path object and optional width, stroke and fillcolor

class SVGdraw.**text**(*x=None*, *y=None*, *text=None*, *font_size=None*, *font_family=None*, *text_anchor=None*, *font_style=None*, ***args*)
Bases: *SVGdraw.SVGelement*

t=text(x,y,text,font_size,font_family,**args)

a text element can bge used for displaying text on the screen

class SVGdraw.**textpath**(*link*, *text=None*, ***args*)
Bases: *SVGdraw.SVGelement*

tp=textpath(text,link,**args)

a textpath places a text on a path which is referenced by a link.

class SVGdraw.**pattern**(*x=None*, *y=None*, *width=None*, *height=None*, *patternUnits=None*, ***args*)
Bases: *SVGdraw.SVGelement*

p=pattern(x,y,width,height,patternUnits,**args)

A pattern is used to fill or stroke an object using a pre-defined graphic object which can be replicated ("tiled") at fixed intervals in x and y to cover the areas to be painted.

class SVGdraw.**title**(*text=None*, ***args*)
Bases: *SVGdraw.SVGelement*

t=title(text,**args)

a title is a text element. The text is displayed in the title bar add at least one to the root svg element

class SVGdraw.**description**(*text=None*, ***args*)
Bases: *SVGdraw.SVGelement*

d=description(text,**args)

a description can be added to any element and is used for a tooltip Add this element before adding other elements.

class SVGdraw.**lineargradient**(*x1=None*, *y1=None*, *x2=None*, *y2=None*, *id=None*, ***args*)
Bases: *SVGdraw.SVGelement*

lg=lineargradient(x1,y1,x2,y2,id,**args)

defines a lineargradient using two xy pairs. stop elements van be added to define the gradient colors.

class SVGdraw.**radialgradient**(*cx=None*, *cy=None*, *r=None*, *fx=None*, *fy=None*, *id=None*, ***args*)
Bases: *SVGdraw.SVGelement*

rg=radialgradient(cx,cy,r,fx,fy,id,**args)

defines a radial gradient using a outer circle which are defined by a cx,cy and r and by using a focalpoint. stop elements van be added to define the gradient colors.

class SVGdraw.**stop**(*offset*, *stop_color=None*, ***args*)
Bases: *SVGdraw.SVGelement*

st=stop(offset,stop_color,**args)

Puts a stop color at the specified radius

class SVGdraw.**style**(*type*, *cdata=None*, ***args*)
Bases: *SVGdraw.SVGelement*

st=style(type,cdata=None,**args)

Add a CDATA element to this element for defing in line stylesheets etc..

class SVGdraw.**image**(*url*, *x=None*, *y=None*, *width=None*, *height=None*, ***args*)
Bases: *SVGdraw.SVGelement*

im=image(url,width,height,x,y,**args)

adds an image to the drawing. Supported formats are .png, .jpg and .svg.

class SVGdraw.**cursor**(*url*, ***args*)
Bases: *SVGdraw.SVGelement*

c=cursor(url,**args)

defines a custom cursor for a element or a drawing

class SVGdraw.**marker**(*id=None*, *viewBox=None*, *refx=None*, *refy=None*, *markerWidth=None*, *markerHeight=None*, ***args*)

Bases: *SVGdraw.SVGelement*

m=marker(id,viewbox,refX,refY,markerWidth,markerHeight,**args)

defines a marker which can be used as an endpoint for a line or other pathtypes add an element to it which should be used as a marker.

class SVGdraw.**group**(*id=None*, ***args*)

Bases: *SVGdraw.SVGelement*

g=group(id,**args)

a group is defined by an id and is used to contain elements g.addElement(SVGelement)

class SVGdraw.**symbol**(*id=None*, *viewBox=None*, ***args*)

Bases: *SVGdraw.SVGelement*

sy=symbol(id,viewbox,**args)

defines a symbol which can be used on different places in your graph using the use element. A symbol is not rendered but you can use 'use' elements to display it by referencing its id. sy.addElement(SVGelement)

class SVGdraw.**defs**(***args*)

Bases: *SVGdraw.SVGelement*

d=defs(**args)

container for defining elements

class SVGdraw.**switch**(***args*)

Bases: *SVGdraw.SVGelement*

sw=switch(**args)

Elements added to a switch element which are "switched" by the attributes requiredFeatures, requiredExtensions and systemLanguage. Refer to the SVG specification for details.

class SVGdraw.**use**(*link*, *x=None*, *y=None*, *width=None*, *height=None*, ***args*)

Bases: *SVGdraw.SVGelement*

u=use(link,x,y,width,height,``**args``)

references a symbol by linking to its id and its position, height and width

class SVGdraw.**link**(*link=''*, ***args*)

Bases: *SVGdraw.SVGelement*

a=link(url,``**args``)

a link is defined by a hyperlink. add elements which have to be linked a.addElement(SVGelement)

class SVGdraw.**view**(*id=None*, ***args*)

Bases: *SVGdraw.SVGelement*

v=view(id,``**args``)

a view can be used to create a view with different attributes

class SVGdraw.**script**(*type*, *cdata=None*, ***args*)

Bases: *SVGdraw.SVGelement*

sc=script(type,type,cdata,``**args``)

adds a script element which contains CDATA to the SVG drawing

class SVGdraw.**animate**(*attribute*, *fr=None*, *to=None*, *dur=None*, ***args*)
Bases: *SVGdraw.SVGelement*

an=animate(attribute,from,to,during,``**args``)

animates an attribute.

class SVGdraw.**animateMotion**(*pathdata*, *dur*, ***args*)
Bases: *SVGdraw.SVGelement*

an=animateMotion(pathdata,dur,``**args``)

animates a SVGelement over the given path in dur seconds

class SVGdraw.**animateTransform**(*type=None*, *fr=None*, *to=None*, *dur=None*, ***args*)
Bases: *SVGdraw.SVGelement*

antr=animateTransform(type,from,to,dur,``**args``)

transform an element from and to a value.

class SVGdraw.**animateColor**(*attribute*, *type=None*, *fr=None*, *to=None*, *dur=None*, ***args*)
Bases: *SVGdraw.SVGelement*

ac=animateColor(attribute,type,from,to,dur,``**args``)

Animates the color of a element

class SVGdraw.**set**(*attribute*, *to=None*, *dur=None*, ***args*)
Bases: *SVGdraw.SVGelement*

st=set(attribute,to,during,``**args``)

sets an attribute to a value for a

class SVGdraw.**svg**(*viewBox=None*, *width=None*, *height=None*, ***args*)
Bases: *SVGdraw.SVGelement*

s=svg(viewbox,width,height,``**args``)

a svg or element is the root of a drawing add all elements to a svg element. You can have different svg elements in one svg file s.addElement(SVGelement)

eg d=drawing() s=svg((0,0,100,100),'100%','100%') c=circle(50,50,20) s.addElement(c) d.setSVG(s) d.toXml()

class SVGdraw.**drawing**
Bases: object

d=drawing()

this is the actual SVG document. It needs a svg element as a root. Use the addSVG method to set the svg to the root. Use the toXml method to write the SVG source to the screen or to a file d=drawing() d.addSVG(svg) d.toXml(optionalfilename)

RateEstimation.py - utilities for computing rate estimates for codon models.

Tags Python

RateEstimation.**evaluateCodonPair**(*codon1*, *codon2*)
evaluate differences between codon pair.

RateEstimation.**countSubstitutions**(*pi*, *Q*)
count substitituions given a matrix Q and frequencies pi.

RateEstimation.**initializeQMatrix**(*codons*)
get an initialized Q matrix.

RateEstimation.**getQMatrix**(*pi*, *Rsi*, *Rsv*, *Rni*, *Rnv*)
build a q matrix.

Diagonal elements are set to the negative of the row sums. The matrix is normalized such that trace of the matrix is -1.

RateEstimation.**getRateMatrix**(*trained_model*, *terminals=None*)
return a rate matrix from an xrate grammar.

terminals: return rate matrix and frequencies for these terminals. If none are given, a dictionaries of matrices and frequencies are returned.

RateEstimation.**setFrequencies**(*model*, *mali*, *prefix=''*)
set frequencies in a model according to those observed in data.

prefix: prefix for rate parameters.

Frequencies are labelled: pa0, pc0, ..., pa1, pc1, ..., pa2, pc2, ...

RateEstimation.**getDistanceGTR**(*pi*, *matrix*)
obtain distance from a GTR model. see Felsenstein 1994, pp 209

Unsorted

Modules not sorted into categories.

7.2.3 Glossary

File formats

yaml Language to serialize objects. Used in the CGAT testing framework. (YAML).

bam Format to store genomic alignments in a compressed format. (BAM).

bed File containing genomic intervals. (BED).

vcf Variant call format.

gtf General transfer format. Format to store genes and transcripts.

gff General feature format.

bigwig Compressed format for displaying numerical values across genomic ranges (BIGWIG).

fasta Sequence format.

wiggle Format for displaying numerical values across genomic ranges (Wiggle).

psl Genomic alignment format. The format is described in detail (PSL.

sam Format to store genomic alignments (SAM).

gdl gdl

tsv Tab separated values. In these tables, records are separated by new-line characters and fields by tab characters. Lines with comments are started by the # character and are ignored. The first uncommented line should contain the column headers. For example:

```
# This is a comment
gene_id       length
gene1 1000
gene2 2000
# Another comment
```

svg pass

edge list pass

fastq Sequence format containing quality scores, more background is here

sra sra

axt axt

agp AGP format

rdf Resource description framework

Other terms

test directory Directory that contains the `test.yaml`, input and reference files for testing scripts.

experiment experiment

replicate replicate

graph graph

track track

graph graph

submit host pass

execution host pass

edge list pass

task pass

sphinxreport sphinxreport

query pass

target pass

code directory pass

go pass

goslim pass

fastq pass

tss Transcription start site

production pipeline A pipeline that performs common tasks on a certain type of data. The idea of a production pipeline is to provide common preprocessing of data and a first look. A *project pipeline* might then take data from one or more *production pipeline* to glean biological insight.

project pipeline A pipeline that is project specific. Usually code is developed first inside a project pipeline. When it becomes generally useful, it may be refactored into a production pipeline.

stdin Unix standard input. Most CGAT tools read data from stdin.

stdout Unix standard output. Most CGAT tools output data to stdout.

stderr Unix standard error. This is where errors go.

loglevel Verbosity of logging information. The logging level can be determined by the `--verbose` option. A level of `0` means no logging output, while `1` is information messages only, while `2` outputs also debugging information.

7.2.4 Dependency graph

This page contains a graph visualization of the dependencies within the CGAT code collection (including the CGAT pipelines). Scripts, modules and pipelines are coloured differently.

CHAPTER

EIGHT

RELEASE NOTES

Notes on each release are below.

8.1 Release 0.4.0

- contributions by Genomics PLC ; https://github.com/cgat-developers/cgat-apps/pull/2
- update installation and conda environments ; https://github.com/cgat-developers/cgat-apps/pull/4
- updated conda environments and README ; https://github.com/cgat-developers/cgat-apps/pull/5
- update conda from 4.3 to 4.5 (solving "CXXABI_1.3.9 not found" error ; https://github.com/ContinuumIO/anaconda-issues/issues/5191) ; https://github.com/cgat-developers/cgat-apps/compare/b1bf0298f984...1f3ebb10ec9b
- new way of activating conda environments ; https://github.com/cgat-developers/cgat-apps/pull/9

CHAPTER

NINE

QUICKSTART

Please install the CGAT-apps using the following *Installation instructions* for dependencies and troubleshooting.

CGAT-apps are run from the unix command line. Lets assume we have the results of the binding locations of a ChIP-Seq experiment (`chipseq.hg19.bed`) in bed format and we want to know, how many binding locations are intronic, intergenic and within exons.

Thus, we need to create a set of genomic annotations denoting intronic, intergenic regions, etc. with respect to a reference gene set. Here, we download the GENCODE geneset (Harrow et al., 2012) in GTF format from ENSEMBL (Flicek et al., 2013).

The following unix statement downloads the ENSEMBL gene set containing over-lapping transcripts, and outputs a set of non-overlapping genomic annotations in gff format (`annotations.gff`) by piping the data through various CGAT tools:

```
wget ftp://ftp.ensembl.org/pub/release-72/gtf/homo_sapiens/Homo_sapiens.GRCh37.72.gtf.
↪gz
| gunzip
| awk '$2 == "protein_coding"'
| cgat gff2ff --genome-file=hg19 --method=sanitize --skip-missing
| cgat gtf2gtf --method=sort --sort-order=gene
| cgat gtf2gtf --method=merge-exons --with-utr
| cgat gtf2gtf --method=filter --filter-method=longest-gene
| cgat gtf2gtf --method=sort --sort-order=position
| cgat gtf2gff --genome-file=hg19 --flank-size=5000 --method=genome
| gzip
> annotations.gff.gz
```

Note: The statements above need an indexed genome. To create such an indexed genome for hg19, type the following:

```
wget http://hgdownload.cse.ucsc.edu/goldenPath/hg19/bigZips/chromFa.tar.gz
| index_fasta.py hg19 - > hg19.log
```

CGAT-apps can be chained into a single work flow using unix pipes. The above sequence of commands in turn (1) reconciles UCSC and ENSEMBL naming schemes for chromosome names, (2) merges all exons of alternative transcripts per gene, (3) keeps the longest gene in case of overlapping genes and (4) annotates exonic, intronic, intergenic and flanking region (size=5kb) within and between genes.

Note that the creation of `annotations.gff.gz` goes beyond simple interval intersection, as gene structures have to be normalized from multiple possible alternative transcripts to a single transcript that is chosen by the user depending on what is most relevant for the analysis.

Choosing different options can provide different sets of answers. Instead of merging all exons per gene, the longest transcript might be selected by replacing (2) with `gtf2gtf --method=filter`

`--filter-method=longest-transcript`. Or, instead of genomic annotations, regulatory domains such as defined by GREAT might be obtained by removing (3) and replacing (4) with `gtf2gff --method=great-domains`.

The generated annotations in annotations.gff can then be used to count the number of transcription factor binding sites using bed-tools or other interval intersections. Here, we will use another CGAT tool, `gtf2table`, to do the counting and classification:

```
zcat /ifs/devel/gat/tutorial/data/srf.hg19.bed
| cgat bed2gff --as-gtf
| cgat gtf2table --counter=classifier-chipseq --gff-file=annotations.gff.gz
```

The scripts follow a consistent naming scheme centered around common genomic formats. Because of the common genomic formats, the tools can be easily combined with other tools such as bedtools (Quinlan and Hall, 2010) or UCSC Tools (Kuhn et al. 2013).

CHAPTER

TEN

DEVELOPER'S GUIDE

10.1 Testing

This module describes the implementation of tests for the CGAT code collection. The CGAT testing includes

- Regression testing of CGAT scripts
- Testing CGAT code for style conformance to pep8
- Testing CGAT code to be importable
- Unit testing of CGAT modules

10.1.1 Regression testing of scripts

Scripts are regression tested by comparing the expected output with the latest output. The tests are implemented in the script `test_scripts.py`.

This script collects tests from subdirectories in the `tests` directory. Each test is named by the name of the script it tests.

Adding a new test manually

To add a new test for a CGAT script, create a new *test directory* in the directory `tests`. The name of the *test directory* has to correspond to the name of the script the tests will tested.

In this directory, create a file called `tests.yaml`. This file is in *yaml* format, a simple text-based format to describe nested data structures.

The `tests.yaml` file contains the descriptions of the individual tests to run. Each test is a separate data structure in this file. The fields are:

options Command line options for running the test. If you need to provide additional files as input, use the `%DIR%` place holder for the *test directory*.

stdin Filename of file to use as stdin to the script. If no stdin is required, set to `null` or omit.

outputs A list of output files obtained by running the script that should be compared to the list of files in `references`. `stdout` signifies the standard output.

references A list of expected output files. The order of `outputs` and `references` should be the same. The reference files are expected to be found in the directory *test directory* and thus need not prefixed with a directory place holder.

description A description of test.

To illustrate, we will be creating tests for the scripts `fasta2counts.py`. First we create the *test directory* `tests/fasta2counts.py`. Next we create a file `tests/fasta2counts.py/tests.yaml` with the following content:

```
basic_test:
    outputs: [stdout]
    stdin: null
    references: [test1.tsv]
    options: --genome-file=<DIR>/small_genome
```

`basic_test` is the name of the test. There is no standard input and the output of the script goes to stdout. Stdout will be compared to the file `test1.tsv`. The script requires the `--genome-file` option, which we supply in the `options` field. The `<DIR>` prefix will be expanded to the directory that contains the file `tests.yaml`.

Finally, we create the required input and reference files in the *test directory*. Our directory structure looks thus:

```
|___tests
  |___fasta2counts.py
  | |___small_genome.fasta
  | |___small_genome.idx
  | |___test1.tsv
  | |___tests.yaml
```

Multiple tests per script can be defined by adding additional data structures in the `tests.yaml` file.

Please write abundant tests, but keep test data to a minimum. Thus, instead of running on a large bam file, create stripped down versions containing only relevant data that is sufficient for the test at hand.

Re-use test data as much as possible. Some generic test data used by multiple tests is in the `tests/data` directory.

Creating a test

The script `tests/setup_test.py` can be used to set up a testing stub. For example:

```
python tests/setup_test.py scripts/bam2bam.py
```

will add a new test for the script `bam2bam.py`.

The script will create a new testing directory for each script passed on the command line and create a simple `tests.yaml` file. The basic test will simply call a script to check if starts without error and returns a version string.

Running tests

The CGAT code collection runs both under nose or py.test. In order to run the tests on CGAT scripts under the nose framework, type:

```
nosetests tests/test_scripts.py
```

In order to get more information, type:

```
nosetests -v tests/test_scripts.py
```

To run individual tests, edit the file `tests/test_scripts.yaml`. In order to restrict testing to a single script, for example `beds2counts.py`, add the following:

```
restrict:
      regex: beds2counts.py
```

To run the tests using py.test, type:

```
py.test tests/test_scripts.py
```

10.1.2 Testing for style

All of CGAT python code are tested for pep8 conformance using the pep8 tools. Not all pep8 rules are enforced, though we aim for increasing compatibility with pep8. Please see also the *Style Guide*.

The testing is controlled by the script `test_style.py`. In order to run the tests, type:

```
nosetests tests/test_scripts.py
```

We have also added a test that will scan all the command line options used in CGAT script against a white- and black-list of acceptable/unacceptable option names. The purpose of this test is to ensure consistency between scripts. To run this test, type:

```
nosetests tests/test_commandline.py
```

This test is based on a list of acceptable/unacceptable options in `tests/option_list.tsv` that is within the repository. The list has been created by the script `cgat_get_option_list.py` and been manually annotated. Errors are flagged if a deprecated option is used in a script or an unregistered option is encountered. To update option list, type:

```
python scripts/cgat_get_option_list.py --in-place --options-tsv-file=tests/option_
↪list.tsv
```

10.1.3 Testing for import

In order for documentation to be built or scripts to be usable by the `cgat` frontend, scripts need to be importable from anywhere. Importability might fail if a script or module executes statements on improt or rely reading from input or configuration files that are not present or have non-sensical values.

The testing is controlled by the script `test_style.py`. In order to run the tests, type:

```
nosetests tests/test_import.py
```

10.1.4 Testing modules

There are some unit tests for specific functions in modules, but a testing regime has not been formalized.

10.1.5 Code coverage

Code coverage for modules can be computed if the python coverage module has been installed. To compute coverage, use:

```
nosetests --with-coverage --cover-package=CGAT --cover-package=scripts tests/test_
↪scripts.py
```

or using py.test:

```
py.test -s tests/test_scripts.py --cov=`pwd` >& out
```

10.2 Style Guide

10.2.1 Coding style

This style guide lays down coding conventions in the CGAT repository. For new scripts, follow the guidelines below.

As the repository has grown over years and several people contributed, the style between scripts can vary. For older scripts, follow the style within a script/module. If you want to apply the newer style, make consistent changes across the script.

In general, we want to adhere to the following conventions:

- *Variable names* are lower case throughout with underscores to separate words, such as `peaks_in_interval = 0`
- ***Function names* start with a lower case character and a** verb. Additional words start in upper case, such as `doSomethingWithData()`
- ***Class names* start with an upper case character, additional words** start again in upper case, such as `class AFancyClass():`
- ***Class methods* follow the same convention as functions, such as** `self.calculateFactor()`
- ***Class attributes* follow the same convention as variables, such** as `self.factor`
- ***Global variables* - in the rare cases they are used, are upper case** throughout such as `DEBUG=False`
- ***Module names* should start with an uppercase letter, for example,** `TreeTools.py` in order to distinguish them from built-in and third-party python modules.
- *Script names* are lower-case throughout with underscores to separate words, for example, `bam2geneprofile.py` or `join_table.py`.
- *Cython extensions* to scripts (via pyximport) should be put into the script name starting with an underscore. For example, The extensions to `bam2geneprofile.py` are in `_bam2geneprofile.pyx`.

For new scripts, use the template `script_template.py`.

The general rule is to write easily readable and maintainable code. Thus, please

- document code liberally and accurately
- **make use of whitespaces and line-breaks to break long statements** into easily readable statements.

In case of uncertainty, follow the python style guides as much as possible. The relevant documents are:

- PEP0008 - Style Guide for Python Code
- PEP0257 - Docstring Conventions

For documenting CGAT code, we follow the conventions for documenting python code:

- Python Developer's guide

For writing doc-strings, we use the numpy guide:

- A guide to Numpy/scipy documentation

See here for an example.

In terms of writing scripts, we follow the following conventions:

- Each script should define the `-h` and `--help` options to give command line help usage.
- For tabular output, scripts should output *tsv* formatted tables. In these tables, records are separated by new-line characters and fields by tab characters. Lines with comments are started by the # character and are ignored. The first uncommented line should contain the column headers. For example:

```
# This is a comment
gene_id length
gene1   1000
gene2   2000
# Another comment
```

- Scripts should follow the unix philosophy. They should concentrate on one task and do it well. Ideally, the major input and output can be read from and written to standard input and standard output, respectively.
- The names of scripts should be meaningful. Most of our scripts perform data transformation of one kind of another, these are often called `a2b.py`. The distinctions can be subtle. Examples are:

  gtf2gtf.py - manipulate transcript models Input is *gtf*, output is *gtf*. This script manipulates gene sets (filtering, merging, ...).

  gtf2gff.py - convert a transcript set to genomic features Input is *gtf*, output is *gff*. This script takes gene sets and changes the hierarchical description within a *gtf* file to the flat description of features in a *gff* file. For example, this script can define gene territories, regulatory domains or genomic annotations based on a gene set.

  bed2gff.py - convert bed to gff/gtf Input is *bed*, output is *gff*. As both formats describe intervals in the genome, this script basically does a conversion between the two formats.

  Quite a few scripts contain the `2table` or `2stats`. These compute, respectively, properties or summary statistics for entries in a file. For example:

  <no title> Input is *gtf*. For each gene or transcript, compute selected properties. If there are 10,000 genes in the input, the output table will contain 10,000 rows.

  gff2stats.py - count features, etc. in gff file Input is *gff*. Compute summary statistics across all features in the file. Here, aggregate sizes or similar by feature type or name per chromosome. No matter if there are 10,000 or 100,000 interval is the input, the output will be have the same number of rows.

10.2.2 Where to put code

Different parts of the code base go into separate directories.

Scripts Scripts are python code that contains a main() function and are intended to be executed. Scripts go into the directory `/scripts`

Modules Modules contain supporting code and are imported by scripts or other modules. Modules go into the directory `/CGAT`.

Pipelines Pipeline scripts and modules go into the directory `/CGATPipelines`.

10.2.3 Pipelines

All components of a pipeline should go into the `CGATPipelines` directory. The basic layout of a pipeline is:

```
CGATPipelines/pipeline_example.py
              /PipelineExample.py
              /PipelineExample.R
              /pipeline_example/pipeline.ini
                               /conf.py
                               /sphinxreport.ini
```

pipeline_example.py The main pipeline code. Pipelines start with the word `pipeline` and follow the conventions for *script names*, all lower case with underscores separating words.

pipeline_example/pipeline.ini Default values for pipeline configuration values.

pipeline_example/conf.py Configuration script for sphinxreport.

pipeline_example/sphinxreport.ini Configuration script for sphinxreport.

pipeline_docs/pipeline_example Sphinxreport for pipeline.

PipelineExample.py Python utility methods and classes specific to this pipeline. Once methods and classes are shared between pipelines, consider moving them to a separate module.

PipelineExample.R R utility functions specific to this pipeline.

- Make sure that the pipeline.ini file exists and contains example/default values with annotation.
- Make sure that the pipeline can be imported from any directory, especially those not containing any data files or configuration files. This is important for the documentation of the pipeline to be built.

10.2.4 Other guidelines

- Only add source code and required data to the repository. Do not add .pyc files, backup files created by your editor or other files.
- In order to build documentation, each script, module and pipeline needs to be importable. Thus, make sure that when your pipeline depends on specific files, it does not fail when imported but not executed.
- There is a style guide for naming script options based on 5 groups. These are designed to increase clarity and familiarity across the script collection.

10.2.5 Script options

The purpose of this section of the style guide is to standardise many of the common options that the CGAT script collection uses. This will add transparency and improve user-friendliness by adding a level of familiarity across scripts.

There are four option groups defined in the guide. Not all options will fit into one of these as many options are specific to a script. This guide will also be a useful reference for new script development by providing a common framework.

The general structure for option names is multiple parts with parts separated by `-`. Generally, aim to have the most significant bit first in the option as option names can be shortened on the command line if they are unambiguous. For example, `--annotation-gtf-file` can be abbreviated as `--annotation-gtf`, `--annotation`, etc.

Single-part options such as `--colours` are permitted if they are unambiguous in the context of the script.

In general single letter (`-a`, `-g`, `...`) type options can be used for very common options, but every option should have a long name and use of long names is preferred in pipelines. If possible, use short letters that are consistent with "related" unix commands.

Option nomenclature that does not fit into one of the below groups should be explicit. For instance use `--output-with-value` instead of `--with-value`.

Option groups: option components in '[]' are variable

- files:

  The file options support both input and ancillary files for scripts. Some scripts require multiple files of the same format. In these instances `[purpose]` differentiates the different files within the script. `--[purpose]-[format]-file=[file]` e.g. `--annotation-gtf-file` or `--bam-file` or `--exons-gtf-file`.

- actions:

  Actions denote the central methods a script applies to the data set. Some scripts might only be able to apply a single action to a data set, while others might allow a sequence of actions to be performed. Scripts that support multiple actions should use the `--methods=[action1, action2,...]`, for example `--methods=sort-by-name,filter-by-length`. Scripts that only support a single action use `--method=[action]`, for example: `--method=select-longest-transcript`.

  Arguments that are relevant for a particular action should be easily associated with the action. In the example above, the minimum length could be given as `--filter-min-length`.

  Do not hesitate to make arguments as explicit as possible. Consider also using: `--method=filter-by-sequence-length` and `--filter-min-sequence-length`.

- parameters:

  Parameters are provided to scripts with a specific purpose. To make these as explicit as possible these also conform to the three-part naming convention. Very common is to set minimum/maximum values. For these, follow a `--[object]-[attribute]-[stat]=[value]` convention, e.g., `--insert-size-min=100` or `--insert-size-std=20`.

- outputs:

  The prinicipal output of a script is generally fed to standard output. Scripts that create multiple output files should define them using the generic `--output-filename-pattern`. Any `%s` pattern will be substituted inside the script with a section name. Optionally, it might also append a suffix for the file type. For example a script called with `--output-filename-pattern="test_%s"` might create files such as `test_plot.png, test_removed.tsv` for the sections `plot` and `removed`.

  In order to facilicate the incorporation of multiple-output scripts into pipelines, scripts should permit explicit labeling of output files such as `--output-filename-<section>` where section corresponds to the sections used in the script. In the example above, the script should also accept options called `--output-filename-plot` and `--output-filename-removed`.

  Note: TODO: This can be implemented generically in Experiment.py

10.2.6 Documentation

Writing doc-strings

Functions should be documented through their doc-string using restructured text. For example:

```
    def computeValue(name, method, accuracy=2):

        :param name: The name to use.
        :type name: str.
        :param method: method to use.
        :type state: choice of ('empirical', 'parametric')
        :param accuracy:
        :type accuracy: integer
        :returns:  int -- the value
        :raises: AttributeError, KeyError

Writing documentation for scripts
---------------------------------

There is a minimum standard for documentation to maintain clarity of
tools and code.  The documentation for any given script should follow
the basic outline in :doc:`scripts/cgat_script_template`.

Three main headers exist:

`Purpose`
   Describe the overall purpose and function of the script and the
   input and output formats.  This can be extensive and include
   sub-headers to further describe script functionality.  For
   example::

       Purpose
       -------

       This script takes a :term:`gtf` formatted file and computes
       meta-gene profiles over various annotations derived from the
       :term:`gtf` file.

       A meta-gene profile is an abstract genomic entity over which
       reads stored in a :term:`bam` formatted file have been
       counted. A meta-gene might be an idealized eukaryotic gene
       (upstream, exonic sequence, downstream) or any other genomic
       landmark of interest such as transcription start sites.
```

Usage Describe example use cases for the script with one or more options. In addition provide the head of both example input and example output files. Example input and output:

```
Usage
-----
    samtools view example.bam

    READ1    163    1     13040   15      76M     =       13183   219     ...
    READ1    83     1     13183   7       76M     =       13040   -219    ...
    READ2    147    1     13207   0       76M     =       13120   -163    ...

    python bam2bed.py example.bam
```

(continues on next page)

(continued from previous page)

```
    1       13039   13115   READ1     15      +
    1       13119   13195   READ2     0       +
    1       13182   13258   READ1     7       -
    1       13206   13282   READ2     0       -
```

Example usage:

```
python example_script.py
      --infile=example.bam
      --option1=choice
      --method=method1
```

Options

Describe all of the options for the script. If necessary provide extensive detail of the methods of each option and how they are combined to provide the intended functionality of the script. This should include all *choice* for options with a verbose description of what that *choice* does. For example:

```
Profiles
--------
Different profiles are accessible through the ``--method`` option. Multiple
methods can be applied at the same time. While ``upstream`` and␣
↪``downstream``
typically have a fixed size, the other regions such as ``CDS``, ``UTR`` will␣
↪be
scaled to a common size.

utrprofile
    UPSTREAM - UTR5 - CDS - UTR3 - DOWNSTREAM
    gene models with UTR. Separate the coding section from the non-coding␣
↪part.
```

There is a fourth template-specific header; the command line options that are automatically generated for every CGAT script:

Command line options These are automatically generated from *cgat_script_template.py - template for cgat scripts* and detail each option specified within the script. No further details need to be added to this section.

In addition, please pay attention to the following:

- Declare input data types for genomic data sets in optparse using the *metavar* keyword. For example:

```
parser.add_option( "--extra-intervals", dest = "extra_intervals",
                   metavar="bed", help = "..." )
```

  Setting the *type* permits the script to be integrated into workflow systems such as galaxy.

- Please provide a meaningful example in the command line help (see above for minimum requirements).
- Be verbose. Something that is not documented within a script will not be used.
- Add meaningful tags to your scripts (`:Tags:`) so that they can be grouped into categories. Please choose from the following controlled vocabulary. If needed, additional terms can be added to this list.
  - Broad Themes
    - Genomics
    - NGS

    * MultipleAlignment
    * GenomeAlignment
    * Intervals
    * Genesets
    * Sequences
    * Variants
    * Protein
- Formats
    * BAM
    * BED
    * GFF
    * GTF
    * FASTA
    * FASTQ
    * WIGGLE
    * PSL
    * CHAIN
- Actions
    * Summary - summarizing entities within a file, such as counting the number of intervals within a file, etc.
    * Annotation - annotating individual entities within a file, such as adding length, composition, etc. to intervals.
    * Comparison - comparing the same type of entities, such as overlapping to sets of intervals.
    * Conversion - converting between different formats for the similar types of objects (Intervals in gff/bed format).
    * Transformation - transforming one entity into another, such as transforming intervals into sequences.
    * Manipulation - changing entities within a file, such as filtering sequences.

10.3 Documentation

10.3.1 Overview

cgat.tools.and modules use sphinx for documentation. The philosophy is to maintain documentation and code together. Thus, most documentation will be kept inside the actual scripts and modules, supported by overview documents explaining usage and higher level concepts. See the *Style Guide* on how to write documentation.

10.3.2 Building the documentation

CGAT's documentation lives in the `doc` directory of the repository. To build the documentation, enter the `doc` directory and type:

```
make html
```

The output will be in the directory `_build/html`.

Note: Each script, module and pipeline needs to be importable, i.e, the following must work:

```
python -c "import pipeline_mapping"
```

Especially in pipelines some care is necessary to avoid failing with an error if no input or configuration files are present.

The page coverage page lists undocumented functions and classes. To update this information, you must set the `COMPUTE_COVERAGE` variable when building the documentation:

```
make html COMPUTE_COVERAGE=1
```

10.3.3 Writing documentation

sphinx documentation is written in Restructured Text. A useful primer is here.

Some specifics for the CGAT code base are:

- Refering to a separate script can be done using the `:doc:` directive, for example:

  ```
  :doc:`scripts/bed2summary`
  ```

  Note that the path relative to the current directory needs to be supplied.

- Glossary terms (`:term:`) are defined in `glossary.rst`.

10.3.4 Adding documentation

In order to add a new script, module or pipeline to the documentation documement, perform the following steps.

Here, we will be adding the script `bed2summary.py` to the documentation.

1. Create a file `doc/scripts/bed2summary.rst` with the following contents:

   ```
   .. automodule:: bed2summary

   .. program-output:: python ../scripts/bed2summary.py --help
   ```

   This will build the documentation within the bed2summary script and add the command line help to the document.

2. Add an entry to `doc/scripts.rst`. For example:

   ```
   .. toctree::

      scripts/bed2summary
   ```

   Please add your script to the toctree of an existing group.

3. For scripts that are part of the CGAT code collection, also add an entry into `doc/CGATReference.rst`.

Adding a module or pipeline is similar to adding a script, except that:

1. the `.rst` file should be in `doc/modules` or `doc/pipelines`, respectively.
2. The entry needs to be added to `modules.rst` or `CGATPipelines.rst`, respectively.
3. no `program-output` is necessary.

10.3.5 Requisites

Building the documentation requires the following components:

sphinx The documenation building system.

sphinxcontrib-programoutput Adding command line output to documenation.

10.3.6 Trouble-shooting

The build may fail with the following error:

```
ImportError: Building module CGAT.NCL.cnestedlist failed: ['ImportError: /ifs/home/
↪XXX/.pyxbld/lib.linux-x86_64-2.7/CGAT/NCL/cnestedlist.so: undefined symbol:␣
↪interval_iterator_alloc\n']
```

In this case, remove the directory `/ifs/home/XXX/.pyxbld/` and restart building the documentation:

```
rm -rf /ifs/home/andreas/.pyxbld/
make html
```

10.4 Importing CGAT scripts into galaxy

10.4.1 General Preparation

Add `/ifs/devel/cgat` to `PYTHONPATH`.

Make sure that extensions have been built:

```
python setup.py develop --multi-version
```

The following directories are important:

galaxy-dist Location of the galaxy distribution

cgat-xml CGAT directory within the galaxy distribution. Create by typing:

```
mkdir <galaxy-dist>/tools/cgat
```

cgat-scripts The CGAT scripts directory.

10.4.2 Adding a script manually

The following instructions describe the steps necessary to add a cgat script to galaxy.

For example, we want to publish the `bam2stats.py` script. First, create a file in `<galaxy-dist>/tools/cgat` called `bam2stats.xml` with the following contents:

```
<tool id="bam2stats.py" name="Compute Stats from BAM file">
  <description>Compute stats for a bam file</description>
    <command
      interpreter="python">/ifs/devel/cgat/scripts/bam2stats.py -v 0 &lt; $input &gt;
↪$output
    </command>
    <inputs>
       <param format="bam" name="input" type="data" label="Source file"/>
    </inputs>
    <outputs>
       <data format="tabular" name="output" />
    </outputs>
  <help>
  Compute statistics for a bam file.
  </help>
</tool>
```

Add an entry to `tool_conf.xml` for the script:

```
<section name="CGAT Tools" id="cgat_tools">
  <tool file="cgat/bam2stats.xml" />
</section>
```

After restarting galaxy, the `bam2stats` command should now be visible in the `CGAT` section.

10.4.3 Automatic conversion of scripts

The CGAT tool collection contains a script called *<no title>* that can create and xml file for inclusion into galaxy. To create a wrapper for *Purpose*, run:

```
python <cgat-scripts>cgat2rdf.py --format=galaxy <cgat-scripts>bam2stats.py > <cgat-
↪xml>bam2stats.xml
```

As before, add an entry to `tool_conf.xml` for the script.

For automatted conversion, a few rules need to be followed (see below).

Writing galaxy compatible scripts

CGAT scripts have generally a call interface that is compatible with galaxy and can thus be easily integrated. However, to make automatic conversion as easy as possible, conforming to a few coding conventions help.

1. Assign a metavar type to command line options of genomic file formats. For example:

```
parser.add_option("-b", "--bam-file", dest="bam_files", type="string", metavar=
↪"bam",
                  help="filename with read mapping"
                       " information. Multiple files can be "
                       " submitted in a comma-separated list"  )
```

2. Use Experiment.OptionParser instead of optparse.OptionParser. The former has some extensions that make creating galaxy xml files easier. In particular, Experiment.OptionParser permits supplying a list of ‘,’-separated values to options that accept multiple values.

3. Follow the CGAT script naming convention. If possible, scripts should be named `<format_in>2<format_out>.py`. Formats can be mapped to other types in *<no title>*. For example, `stats` and `table` are both mapped to the format `tabular`.

CHAPTER

ELEVEN

GLOSSARY

CHAPTER

TWELVE

DISCLAIMER

This collection of scripts is the outcome of 10 years working in various fields in bioinformatics. It contains both the good, the bad and the ugly. Use at your own risk.

PYTHON MODULE INDEX

a

AGP, 164
AString, 190

b

BamTools, 165
Bed, 165
Blat, 168

c

CBioPortal, 170
cgat.tools.bam2bam, 89
cgat.tools.bam2bed, 91
cgat.tools.bam2fastq, 93
cgat.tools.bam2geneprofile, 21
cgat.tools.bam2peakshape, 94
cgat.tools.bam2stats, 97
cgat.tools.bam2UniquePairs, 88
cgat.tools.bam2wiggle, 59
cgat.tools.bam_vs_bam, 124
cgat.tools.bam_vs_bed, 125
cgat.tools.bams2bam, 144
cgat.tools.bed2annotator, 60
cgat.tools.bed2bed, 26
cgat.tools.bed2fasta, 103
cgat.tools.bed2gff, 29
cgat.tools.bed2graph, 61
cgat.tools.bed2plot, 145
cgat.tools.bed2stats, 104
cgat.tools.beds2beds, 106
cgat.tools.beds2counts, 101
cgat.tools.cat_tables, 78
cgat.tools.cgat2dot, 147
cgat.tools.cgat_galaxy_wrapper, 148
cgat.tools.cgat_get_options, 148
cgat.tools.cgat_pep8_code_quality, 149
cgat.tools.cgat_rebuild_extensions, 120
cgat.tools.cgat_script_template, 87
cgat.tools.chain2psl, 61
cgat.tools.combine_tables, 108
cgat.tools.csv2csv, 73
cgat.tools.csv2db, 73
cgat.tools.csv_cut, 75
cgat.tools.csv_intersection, 75
cgat.tools.csv_rename, 76
cgat.tools.csv_select, 151
cgat.tools.csv_set, 77
cgat.tools.csvs2csv, 72
cgat.tools.data2histogram, 83
cgat.tools.diff_bam, 128
cgat.tools.diff_bed, 62
cgat.tools.diff_chains, 109
cgat.tools.diff_fasta, 70
cgat.tools.diff_gtf, 32
cgat.tools.fasta2bed, 63
cgat.tools.fasta2fasta, 129
cgat.tools.fasta2kmercontent, 131
cgat.tools.fasta2variants, 111
cgat.tools.fastas2fasta, 133
cgat.tools.fastq2fastq, 112
cgat.tools.fastq2summary, 154
cgat.tools.fastq2table, 115
cgat.tools.fastqs2fasta, 134
cgat.tools.fastqs2fastq, 155
cgat.tools.fastqs2fastqs, 135
cgat.tools.genome_bed, 117
cgat.tools.gff2bed, 30
cgat.tools.gff2coverage, 36
cgat.tools.gff2fasta, 37
cgat.tools.gff2gff, 39
cgat.tools.gff2histogram, 42
cgat.tools.gff2psl, 35
cgat.tools.gff2stats, 44
cgat.tools.gff2table, 64
cgat.tools.gff32gtf, 156
cgat.tools.gtf2fasta, 55
cgat.tools.gtf2gff, 45
cgat.tools.gtf2gtf, 50
cgat.tools.gtf2tsv, 137
cgat.tools.gtfs2tsv, 139
cgat.tools.index2bed, 118
cgat.tools.index_fasta, 68
cgat.tools.medip_merge_intervals, 119
cgat.tools.metaphlan2table, 158

cgat.tools.rnaseq_junction_bam2bam, 141
cgat.tools.split_fasta, 160
cgat.tools.split_file, 86
cgat.tools.split_gff, 142
cgat.tools.table2table, 79
cgat.tools.transfac2transfac, 162
cgat.tools.vcf2vcf, 122
cgat.tools.vcfstats2db, 123
cgat.tools.wig2bed, 163

f

FastaIterator, 175
Fastq, 176

g

Genomics, 190
GFF3, 179
GTF, 179

h

Histogram, 211
Histogram2D, 214

i

IGV, 209
IndexedFasta, 184
IndexedGenome, 187
Intervals, 196
Iterators, 220

m

Masker, 210
MatrixTools, 218
Motifs, 198

r

RateEstimation, 235
RLE, 229

s

SequencePairProperties, 198
SequenceProperties, 199
SetTools, 220
Sra, 189
Stats, 214
SVGdraw, 229

t

tools.bam_vs_gtf, 126
Tree, 222
TreeTools, 224

v

Variants, 205
VCF, 189

w

WrapperCodeML, 206

INDEX

A

Accumulate() (*in module Histogram*), 212
Add() (*in module Histogram*), 212
addAlignments() (*in module Blat*), 170
addComplementIntervals() (*in module Intervals*), 197
addElement() (*SVGdraw.SVGelement method*), 231
addOptions() (*in module MatrixTools*), 218
AddOptions() (*WrapperCodeML.BaseML method*), 208
AddOptions() (*WrapperCodeML.CodeML method*), 207
addProperties() (*SequenceProperties.SequenceProperties method*), 200
addProperties() (*SequenceProperties.SequencePropertiesAA method*), 202
addProperties() (*SequenceProperties.SequencePropertiesAminoAcids method*), 203
addProperties() (*SequenceProperties.SequencePropertiesCodons method*), 203
addProperties() (*SequenceProperties.SequencePropertiesCodonTranslator method*), 203
addProperties() (*SequenceProperties.SequencePropertiesCodonUsage method*), 203
addProperties() (*SequenceProperties.SequencePropertiesCounts method*), 204
addProperties() (*SequenceProperties.SequencePropertiesCpg method*), 201
addProperties() (*SequenceProperties.SequencePropertiesDegeneracy method*), 202
addProperties() (*SequenceProperties.SequencePropertiesDN method*), 201
addProperties() (*SequenceProperties.SequencePropertiesEntropy method*), 205
addProperties() (*SequenceProperties.SequencePropertiesGaps method*), 201
addProperties() (*SequenceProperties.SequencePropertiesLength method*), 200
addProperties() (*SequenceProperties.SequencePropertiesNA method*), 201
addProperties() (*SequenceProperties.SequencePropertiesSequence method*), 200
AddRelativeAndCumulativeDistributions() (*in module Histogram*), 213
adjustPValues() (*in module Stats*), 217
after() (*IndexedGenome.Quicksect method*), 188
AGP
    module, 164
agp, **237**
AGP (*class in AGP*), 164
AlignedPair2SubstitutionMatrix() (*in module Genomics*), 195
Alignment2CDNA() (*in module Genomics*), 194
Alignment2DNA() (*in module Genomics*), 193
Alignment2ExonBoundaries() (*in module Genomics*), 192
Alignment2PeptideAlignment() (*in module Genomics*), 194
Alignment2String() (*in module Genomics*), 192
AlignmentProtein2CDNA() (*in module Genomics*), 194
animate (*class in SVGdraw*), 234
animateColor (*class in SVGdraw*), 235
animateMotion (*class in SVGdraw*), 235
animateTransform (*class in SVGdraw*), 235
asDict() (*GTF.Entry method*), 184
asRanges() (*in module GTF*), 181
AString
    module, 190
AString (*class in AString*), 190
attributes (*GTF.Entry attribute*), 184
axt, **237**

B

bam, **236**

bamG, **144**
bamT, **144**
BamTools
    module, 165
BaseML (*class in WrapperCodeML*), 208
BaseMLResult (*class in WrapperCodeML*), 207
Bed
    module, 165
bed, **236**
Bed (*class in Bed*), 165
bed_iterator() (*in module Bed*), 167
before() (*IndexedGenome.Quicksect method*), 188
benchmarkRandomFragment() (*in module IndexedFasta*), 187
bezier() (*SVGdraw.pathdata method*), 230
bigwig, **236**
binIntervals() (*in module Bed*), 167
bins() (*in module cgat.tools.bed2bed*), 27
Blat
    module, 168
block() (*in module cgat.tools.bed2bed*), 27
blockCount (*Bed.Bed attribute*), 166
blocked_iterator() (*in module Bed*), 167
blockSizes (*Bed.Bed attribute*), 166
blockStarts (*Bed.Bed attribute*), 166
Branchlength2Support() (*in module TreeTools*), 225
buildAlleles() (*in module Variants*), 206
BuildMapSpecies2Genes() (*in module TreeTools*), 225
buildMatrixFromEdges() (*in module MatrixTools*), 219
buildMatrixFromLists() (*in module MatrixTools*), 218
buildMatrixFromTables() (*in module MatrixTools*), 218
buildOffsets() (*in module Variants*), 206
buildSubstitutionMatrix() (*SequencePairProperties.SequencePairPropertiesCountsNa method*), 199

C

Calculate() (*in module Histogram*), 212
Calculate() (*in module Histogram2D*), 214
CalculateCAIWeightsFromCounts() (*in module Genomics*), 195
CalculateCodonFrequenciesFromCounts() (*in module Genomics*), 195
CalculateConst() (*in module Histogram*), 212
CalculateFromTable() (*in module Histogram*), 211
calculateOverlap() (*in module Intervals*), 197
CalculatePairIndices() (*in module Genomics*), 195
calculatePatternsFromTree() (*in module TreeTools*), 229
CalculateRCSUValuesFromCounts() (*in module Genomics*), 195
calculateScale() (*WrapperCodeML.Evolver method*), 208
CBioPortal
    module, 170
CBioPortal (*class in CBioPortal*), 172
CDGSError, 175
cgat.tools.bam2bam
    module, 89
cgat.tools.bam2bed
    module, 91
cgat.tools.bam2fastq
    module, 93
cgat.tools.bam2geneprofile
    module, 21
cgat.tools.bam2peakshape
    module, 94
cgat.tools.bam2stats
    module, 97
cgat.tools.bam2UniquePairs
    module, 88
cgat.tools.bam2wiggle
    module, 59
cgat.tools.bam_vs_bam
    module, 124
cgat.tools.bam_vs_bed
    module, 125
cgat.tools.bams2bam
    module, 144
cgat.tools.bed2annotator
    module, 60
cgat.tools.bed2bed
    module, 26
cgat.tools.bed2fasta
    module, 103
cgat.tools.bed2gff
    module, 29
cgat.tools.bed2graph
    module, 61
cgat.tools.bed2plot
    module, 145
cgat.tools.bed2stats
    module, 104
cgat.tools.beds2beds
    module, 106
cgat.tools.beds2counts
    module, 101
cgat.tools.cat_tables
    module, 78
cgat.tools.cgat2dot
    module, 147

cgat.tools.cgat_galaxy_wrapper
    module, 148
cgat.tools.cgat_get_options
    module, 148
cgat.tools.cgat_pep8_code_quality
    module, 149
cgat.tools.cgat_rebuild_extensions
    module, 120
cgat.tools.cgat_script_template
    module, 87
cgat.tools.chain2psl
    module, 61
cgat.tools.combine_tables
    module, 108
cgat.tools.csv2csv
    module, 73
cgat.tools.csv2db
    module, 73
cgat.tools.csv_cut
    module, 75
cgat.tools.csv_intersection
    module, 75
cgat.tools.csv_rename
    module, 76
cgat.tools.csv_select
    module, 151
cgat.tools.csv_set
    module, 77
cgat.tools.csvs2csv
    module, 72
cgat.tools.data2histogram
    module, 83
cgat.tools.diff_bam
    module, 128
cgat.tools.diff_bed
    module, 62
cgat.tools.diff_chains
    module, 109
cgat.tools.diff_fasta
    module, 70
cgat.tools.diff_gtf
    module, 32
cgat.tools.fasta2bed
    module, 63
cgat.tools.fasta2fasta
    module, 129
cgat.tools.fasta2kmercontent
    module, 131
cgat.tools.fasta2variants
    module, 111
cgat.tools.fastas2fasta
    module, 133
cgat.tools.fastq2fastq
    module, 112
cgat.tools.fastq2summary
    module, 154
cgat.tools.fastq2table
    module, 115
cgat.tools.fastqs2fasta
    module, 134
cgat.tools.fastqs2fastq
    module, 155
cgat.tools.fastqs2fastqs
    module, 135
cgat.tools.genome_bed
    module, 117
cgat.tools.gff2bed
    module, 30
cgat.tools.gff2coverage
    module, 36
cgat.tools.gff2fasta
    module, 37
cgat.tools.gff2gff
    module, 39
cgat.tools.gff2histogram
    module, 42
cgat.tools.gff2psl
    module, 35
cgat.tools.gff2stats
    module, 44
cgat.tools.gff2table
    module, 64
cgat.tools.gff32gtf
    module, 156
cgat.tools.gtf2fasta
    module, 55
cgat.tools.gtf2gff
    module, 45
cgat.tools.gtf2gtf
    module, 50
cgat.tools.gtf2tsv
    module, 137
cgat.tools.gtfs2tsv
    module, 139
cgat.tools.index2bed
    module, 118
cgat.tools.index_fasta
    module, 68
cgat.tools.medip_merge_intervals
    module, 119
cgat.tools.metaphlan2table
    module, 158
cgat.tools.rnaseq_junction_bam2bam
    module, 141
cgat.tools.split_fasta
    module, 160
cgat.tools.split_file
    module, 86

cgat.tools.split_gff
    module, 142
cgat.tools.table2table
    module, 79
cgat.tools.transfac2transfac
    module, 162
cgat.tools.vcf2vcf
    module, 122
cgat.tools.vcfstats2db
    module, 123
cgat.tools.wig2bed
    module, 163
cgatIndexedFasta (*class in IndexedFasta*), 186
checkSection() (*WrapperCodeML.CodeML method*), 207
chrom_iterator() (*in module GFF3*), 179
chunk_iterator() (*in module GTF*), 180
circle (*class in SVGdraw*), 232
clean_cache() (*in module Sra*), 189
closepath() (*SVGdraw.pathdata method*), 230
code directory, **237**
CodeML (*class in WrapperCodeML*), 207
CodeMLAncestralSequence (*class in WrapperCodeML*), 207
CodeMLBranchInfo (*class in WrapperCodeML*), 206
CodeMLPairwise (*class in WrapperCodeML*), 208
CodeMLResultPair (*class in WrapperCodeML*), 207
CodeMLResultPairs (*class in WrapperCodeML*), 207
CodeMLResultSites (*class in WrapperCodeML*), 207
CodeMLSites (*class in WrapperCodeML*), 208
columns() (*Bed.Bed property*), 167
combinations() (*in module SetTools*), 221
Combine() (*in module Histogram*), 212
combine() (*in module Intervals*), 196
combineAtDistance() (*in module Intervals*), 197
CombineOverlaps() (*in module GTF*), 182
compareLists() (*in module SetTools*), 222
complement() (*in module Intervals*), 196
compress() (*in module RLE*), 229
compressIndex() (*IndexedFasta.cgatIndexedFasta method*), 186
computeROC() (*in module Stats*), 216
contig (*Bed.Bed attribute*), 165
contig (*GTF.Entry attribute*), 183
convert() (*in module Histogram*), 212
convertCoordinates() (*Blat.Match method*), 169
convertStrand() (*in module Genomics*), 195
convertTree2Graph() (*in module TreeTools*), 229
copy() (*Bed.Bed method*), 166
copy() (*GTF.Entry method*), 184
CorrelationTest (*class in Stats*), 215
count() (*in module FastaIterator*), 176
Count() (*in module Histogram*), 212
CountBranchPoints() (*in module TreeTools*), 227
CountCodons() (*in module Genomics*), 195
CountDuplications() (*in module TreeTools*), 228
CountGeneFeatures() (*in module Genomics*), 192
countMotifs() (*in module Motifs*), 198
countSubstitutions() (*in module RateEstimation*), 236
createDatabase() (*in module IndexedFasta*), 185
Cumulate() (*in module Histogram*), 213
cumulate() (*in module Histogram*), 213
cursor (*class in SVGdraw*), 233

D

decode() (*in module RLE*), 229
decodeGenotype() (*in module Genomics*), 194
defs (*class in SVGdraw*), 234
description (*class in SVGdraw*), 233
dfs() (*Tree.Tree method*), 223
DistributionalParameters (*class in Stats*), 215
doBinomialTest() (*in module Stats*), 214
doChiSquaredTest() (*in module Stats*), 214
doCorrelationTest() (*in module Stats*), 216
Documentation() (*in module cgat.tools.vcf2vcf*), 122
doFDRPython() (*in module Stats*), 215
doLogLikelihoodTest() (*in module Stats*), 214
doMannWhitneyUTest() (*in module Stats*), 217
doPairedTTest() (*in module Stats*), 216
doPearsonChiSquaredTest() (*in module Stats*), 215
doWelchsTTest() (*in module Stats*), 216
drawing (*class in SVGdraw*), 235

E

edge list, **237**
ellarc() (*SVGdraw.pathdata method*), 231
ellipse (*class in SVGdraw*), 232
encode() (*in module RLE*), 229
encodeGenotype() (*in module Genomics*), 193
end (*Bed.Bed attribute*), 165
end (*GTF.Entry attribute*), 183
Entry (*class in GFF3*), 179
Entry (*class in GTF*), 183
environment variable
    PYTHONPATH, 254
Error, 168, 182, 206
evaluateCodonPair() (*in module RateEstimation*), 236
Evolver (*class in WrapperCodeML*), 208
EvolverBaseml (*class in WrapperCodeML*), 209
execution host, **237**
Exons2Alignment() (*in module Genomics*), 194
experiment, **237**

ExtendedVariant (*in module Variants*), 205
extract() (*in module Sra*), 189

F

fasta, **236**
FastaIterator
    module, 175
FastaIterator (*class in FastaIterator*), 176
FastaRecord (*class in FastaIterator*), 176
Fastq
    module, 176
fastq, **237**
feature (*GTF.Entry attribute*), 183
fetch_ENA() (*in module Sra*), 189
fetch_ENA_files() (*in module Sra*), 189
fetch_TCGA_BAM() (*in module Sra*), 189
fetch_TCGA_fastq() (*in module Sra*), 189
Fill() (*in module Histogram*), 212
fill() (*in module Histogram*), 213
fillHistograms() (*in module Histogram*), 213
filterMasked() (*in module Stats*), 216
flat_file_iterator() (*in module GFF3*), 179
flat_gene_iterator() (*in module GTF*), 181
fn() (*Stats.ROCResult property*), 217
fnr() (*Stats.ROCResult property*), 217
fold (*FastaIterator.FastaRecord attribute*), 176
forceForwardCoordinates() (*in module Genomics*), 191
format (*Fastq.Record attribute*), 177
fp() (*Stats.ROCResult property*), 217
fpr() (*Stats.ROCResult property*), 217
frame (*GTF.Entry attribute*), 183
fromArray() (*in module Intervals*), 197
fromBed() (*GTF.Entry method*), 184
fromGTF() (*Bed.Bed method*), 166
fromGTF() (*GTF.Entry method*), 184
fromIntervals() (*Bed.Bed method*), 166
fromMali() (*WrapperCodeML.Evolver method*), 208
fromMali() (*WrapperCodeML.EvolverBaseml method*), 209
fromMap() (*Blat.Match method*), 169
fromMaq() (*Blat.Match method*), 169
fromPair() (*Blat.Match method*), 169
fromPhred() (*Fastq.Record method*), 177
fromPSL() (*Blat.MatchPSLX method*), 170
fromResult() (*WrapperCodeML.CodeMLResultPairs method*), 207

G

gdl, **237**
gene_id (*GTF.Entry attribute*), 183
gene_iterator() (*in module GTF*), 180
Genes2Species() (*in module TreeTools*), 225
Genomics
    module, 190
genotype() (*Variants.Variant property*), 205
get() (*IndexedGenome.IndexedGenome method*), 188
get() (*IndexedGenome.Quicksect method*), 188
get_leaves() (*Tree.Tree method*), 223
get_nodes() (*Tree.Tree method*), 223
GetAlignmentLength() (*in module Genomics*), 192
getAllCombinations() (*in module SetTools*), 221
GetAllNodes() (*in module TreeTools*), 227
getAlphabet() (*Masker.Masker method*), 211
getAreaUnderCurve() (*in module Stats*), 216
getAttributeField() (*GFF3.Entry method*), 179
GetBiasedCodonUsage() (*in module Genomics*), 195
getBlocks() (*Blat.Match method*), 169
GetBranchLengths() (*in module TreeTools*), 227
getCancerStudies() (*CBioPortal.CBioPortal method*), 172
getCaseLists() (*CBioPortal.CBioPortal method*), 172
GetChildNodesWhereTrue() (*in module TreeTools*), 228
getClinicalData() (*CBioPortal.CBioPortal method*), 173
GetCommonAncestor() (*in module TreeTools*), 226
getComponents() (*in module Blat*), 170
getContigs() (*IndexedFasta.cgatIndexedFasta method*), 186
getContigSizes() (*IndexedFasta.cgatIndexedFasta method*), 186
getConverter() (*in module IndexedFasta*), 187
getCounts() (*in module Motifs*), 198
getDatabaseName() (*IndexedFasta.cgatIndexedFasta method*), 186
GetDegenerateSites() (*in module Genomics*), 194
getDistanceGTR() (*in module RateEstimation*), 236
GetDistancesBetweenTaxa() (*in module TreeTools*), 227
GetDistanceToRoot() (*in module TreeTools*), 229
getEntropy() (*SequenceProperties.SequencePropertiesBias method*), 204
getEntropy() (*SequenceProperties.SequencePropertiesEntropy method*), 205
getGCContent() (*Genomics.SequencePairInfo method*), 195
getGeneticProfiles() (*CBioPortal.CBioPortal method*), 172
getHeader() (*Stats.DistributionalParameters method*), 215
getHeader() (*Stats.Summary method*), 215
getHeaders() (*SequenceProperties.SequencePropertiesAA method*), 202
getHeaders() (*Stats.DistributionalParameters*

method), 215
getHeaders() (Stats.Summary method), 215
GetHID() (in module Genomics), 191
getIntersections() (in module Intervals), 197
GetIntronType() (in module Genomics), 195
getKL() (SequenceProperties.SequencePropertiesBias method), 204
GetLeaves() (in module TreeTools), 226
getLength() (in module Intervals), 196
getLength() (IndexedFasta.cgatIndexedFasta method), 186
getLengths() (IndexedFasta.cgatIndexedFasta method), 186
getLink() (CBioPortal.CBioPortal method), 174
GetMapAA2Codons() (in module Genomics), 194
getMapQuery2Target() (Blat.Match method), 169
getMapTarget2Query() (Blat.Match method), 169
getMatrixFromEdges() (in module MatrixTools), 218
GetMaxIndex() (in module TreeTools), 226
getMessageLength() (SequenceProperties.SequencePropertiesBias method), 204
GetMonophyleticPairs() (in module TreeTools), 225
getMutationData() (CBioPortal.CBioPortal method), 173
GetNodeMap() (in module TreeTools), 227
getNumColumns() (in module Bed), 168
getNumLeaves() (Tree.Tree method), 223
getOccurances() (in module Motifs), 198
getOffset() (in module Fastq), 179
getOncoprintHTML() (CBioPortal.CBioPortal method), 174
getOptions() (in module WrapperCodeML), 209
GetOptions() (WrapperCodeML.CodeML method), 207
getParameters() (WrapperCodeML.EvolverBaseml method), 209
GetParentNodeWhereTrue() (in module TreeTools), 228
getPercentAltered() (CBioPortal.CBioPortal method), 175
getPerformance() (in module Stats), 217
getPooledVariance() (in module Stats), 216
getProfileData() (CBioPortal.CBioPortal method), 172
getProteinArrayData() (CBioPortal.CBioPortal method), 174
getProteinArrayInfo() (CBioPortal.CBioPortal method), 174
getQMatrix() (in module RateEstimation), 236
getRandomCoordinates() (IndexedFasta.cgatIndexedFasta method), 186
getRateMatrix() (in module RateEstimation), 236
getReadLength() (in module Fastq), 179
getSection() (WrapperCodeML.CodeML method), 207
getSensitivityRecall() (in module Stats), 216
getSequence() (IndexedFasta.cgatIndexedFasta method), 186
getSequence() (IndexedFasta.PysamIndexedFasta method), 186
getSignificance() (in module Stats), 214
GetSize() (in module TreeTools), 227
GetSubsets() (in module TreeTools), 227
GetSubtree() (in module TreeTools), 227
GetTaxa() (in module TreeTools), 224
GetTaxaForSpecies() (in module TreeTools), 225
GetTaxonomicNames() (in module TreeTools), 224
getToken() (IndexedFasta.cgatIndexedFasta method), 186
getTotalAltered() (CBioPortal.CBioPortal method), 175
GetUniformCodonUsage() (in module Genomics), 195
getZScore() (Stats.DistributionalParameters method), 215
gff, **236**
GFF3
    module, 179
go, **237**
goslim, **237**
graph, **237**
Graph2Tree() (in module TreeTools), 227
group (class in SVGdraw), 234
group_by_distance() (in module Iterators), 220
grouped_iterator() (in module Bed), 167
GTF
    module, 179
gtf, **236**
guessDataType() (Fastq.Record method), 177
guessDataType() (in module Fastq), 178
guessFormat() (Fastq.Record method), 177
guessFormat() (in module Fastq), 178

H

HalfIdentity() (in module GTF), 181
hasOverlap() (GTF.Entry method), 184
Histogram
    module, 211
histogram() (in module Histogram), 213
Histogram2D
    module, 214
hline() (SVGdraw.pathdata method), 230

I

identifier (Fastq.Record attribute), 177
Identity() (in module GTF), 181

IGV
    module, 209
IGV (*class in IGV*), 209
image (*class in SVGdraw*), 233
index_factory (*IndexedGenome.IndexedGenome attribute*), 188
index_factory (*IndexedGenome.Simple attribute*), 188
IndexedFasta
    module, 184
IndexedFasta() (*in module IndexedFasta*), 187
IndexedGenome
    module, 187
IndexedGenome (*class in IndexedGenome*), 188
indexVariants() (*in module Variants*), 206
initializeQMatrix() (*in module RateEstimation*), 236
intersect() (*in module Intervals*), 197
Intervals
    module, 196
invert() (*GTF.Entry method*), 184
IsCompatible() (*in module TreeTools*), 227
isHalfIdentical() (*GTF.Entry method*), 184
isIdentical() (*GTF.Entry method*), 184
IsJunk() (*in module Genomics*), 195
IsMonophyleticForSpecies() (*in module TreeTools*), 225
IsMonophyleticForTaxa() (*in module TreeTools*), 226
IsSingleSpecies() (*in module TreeTools*), 226
itemRGB (*Bed.Bed attribute*), 166
iterate() (*in module FastaIterator*), 176
iterate() (*in module Fastq*), 177
iterate_convert() (*in module Fastq*), 178
iterate_guess() (*in module Fastq*), 177
iterate_together() (*in module FastaIterator*), 176
iterator() (*in module Bed*), 167
iterator() (*in module Blat*), 170
iterator() (*in module GTF*), 180
iterator2() (*in module Blat*), 170
iterator_contigs() (*in module GTF*), 180
iterator_filtered() (*in module GTF*), 181
iterator_from_gff() (*in module GFF3*), 179
iterator_min_feature_length() (*in module GTF*), 181
iterator_overlapping_genes() (*in module GTF*), 181
iterator_overlaps() (*in module GTF*), 181
iterator_per_query() (*in module Blat*), 170
iterator_pslx() (*in module Blat*), 170
iterator_query_overlap() (*in module Blat*), 170
iterator_sorted() (*in module GTF*), 181
iterator_sorted_chunks() (*in module GTF*), 181
iterator_target_overlap() (*in module Blat*), 170
iterator_test() (*in module Blat*), 170
iterator_transcripts2genes() (*in module GTF*), 181
Iterators
    module, 220
iupac2regex() (*in module Motifs*), 198

J

joined_iterator() (*in module GTF*), 180
joined_iterator() (*in module Intervals*), 197

L

line (*class in SVGdraw*), 232
line() (*SVGdraw.pathdata method*), 230
lineargradient (*class in SVGdraw*), 233
link (*class in SVGdraw*), 234
loadPair() (*SequencePairProperties.SequencePairPropertiesBaseML method*), 198
loadPair() (*SequencePairProperties.SequencePairPropertiesCountsNa method*), 199
loadPair() (*SequencePairProperties.SequencePairPropertiesPID method*), 199
loadSequence() (*SequenceProperties.SequenceProperties method*), 200
loadSequence() (*SequenceProperties.SequencePropertiesAA method*), 202
loadSequence() (*SequenceProperties.SequencePropertiesAminoAcids method*), 203
loadSequence() (*SequenceProperties.SequencePropertiesCodons method*), 203
loadSequence() (*SequenceProperties.SequencePropertiesCodonTranslator method*), 203
loadSequence() (*SequenceProperties.SequencePropertiesCounts method*), 204
loadSequence() (*SequenceProperties.SequencePropertiesCpg method*), 201
loadSequence() (*SequenceProperties.SequencePropertiesDegeneracy method*), 202
loadSequence() (*SequenceProperties.SequencePropertiesDN method*), 201
loadSequence() (*SequenceProperties.SequencePropertiesGaps method*), 201

loadSequence() (*SequenceProperties.SequencePropertiesHid method*), 200
loadSequence() (*SequenceProperties.SequencePropertiesLength method*), 200
loadSequence() (*SequenceProperties.SequencePropertiesNA method*), 201
loadSequence() (*SequenceProperties.SequencePropertiesSequence method*), 200
loglevel, **238**
lower() (*AString.AString method*), 190

M

makeSubstitutionMatrix() (*in module Genomics*), 195
MapCodon2AA() (*in module Genomics*), 194
mapLocation() (*AGP.AGP method*), 165
MapTaxa() (*in module TreeTools*), 224
MapTerminalTaxa() (*in module TreeTools*), 226
marker (*class in SVGdraw*), 233
Masker
    module, 210
Masker (*class in Masker*), 211
MaskerBias (*class in Masker*), 211
MaskerDustMasker (*class in Masker*), 211
MaskerRandom (*class in Masker*), 211
MaskerSeg (*class in Masker*), 211
maskSequence() (*Masker.Masker method*), 211
maskSequences() (*in module Masker*), 211
maskSequences() (*Masker.Masker method*), 211
MaskStopCodons() (*in module Genomics*), 193
Match (*class in Blat*), 168
MatchPSLX (*class in Blat*), 170
MatrixTools
    module, 218
merge() (*in module Bed*), 168
merge() (*in module cgat.tools.bed2bed*), 26
merged_gene_iterator() (*in module GTF*), 181
mergeVariants() (*in module Variants*), 205
module
    AGP, 164
    AString, 190
    BamTools, 165
    Bed, 165
    Blat, 168
    CBioPortal, 170
    cgat.tools.bam2bam, 89
    cgat.tools.bam2bed, 91
    cgat.tools.bam2fastq, 93
    cgat.tools.bam2geneprofile, 21
    cgat.tools.bam2peakshape, 94
    cgat.tools.bam2stats, 97
    cgat.tools.bam2UniquePairs, 88
    cgat.tools.bam2wiggle, 59
    cgat.tools.bam_vs_bam, 124
    cgat.tools.bam_vs_bed, 125
    cgat.tools.bams2bam, 144
    cgat.tools.bed2annotator, 60
    cgat.tools.bed2bed, 26
    cgat.tools.bed2fasta, 103
    cgat.tools.bed2gff, 29
    cgat.tools.bed2graph, 61
    cgat.tools.bed2plot, 145
    cgat.tools.bed2stats, 104
    cgat.tools.beds2beds, 106
    cgat.tools.beds2counts, 101
    cgat.tools.cat_tables, 78
    cgat.tools.cgat2dot, 147
    cgat.tools.cgat_galaxy_wrapper, 148
    cgat.tools.cgat_get_options, 148
    cgat.tools.cgat_pep8_code_quality, 149
    cgat.tools.cgat_rebuild_extensions, 120
    cgat.tools.cgat_script_template, 87
    cgat.tools.chain2psl, 61
    cgat.tools.combine_tables, 108
    cgat.tools.csv2csv, 73
    cgat.tools.csv2db, 73
    cgat.tools.csv_cut, 75
    cgat.tools.csv_intersection, 75
    cgat.tools.csv_rename, 76
    cgat.tools.csv_select, 151
    cgat.tools.csv_set, 77
    cgat.tools.csvs2csv, 72
    cgat.tools.data2histogram, 83
    cgat.tools.diff_bam, 128
    cgat.tools.diff_bed, 62
    cgat.tools.diff_chains, 109
    cgat.tools.diff_fasta, 70
    cgat.tools.diff_gtf, 32
    cgat.tools.fasta2bed, 63
    cgat.tools.fasta2fasta, 129
    cgat.tools.fasta2kmercontent, 131
    cgat.tools.fasta2variants, 111
    cgat.tools.fastas2fasta, 133
    cgat.tools.fastq2fastq, 112
    cgat.tools.fastq2summary, 154
    cgat.tools.fastq2table, 115
    cgat.tools.fastqs2fasta, 134
    cgat.tools.fastqs2fastq, 155
    cgat.tools.fastqs2fastqs, 135
    cgat.tools.genome_bed, 117
    cgat.tools.gff2bed, 30
    cgat.tools.gff2coverage, 36
    cgat.tools.gff2fasta, 37
    cgat.tools.gff2gff, 39

cgat.tools.gff2histogram, 42
cgat.tools.gff2psl, 35
cgat.tools.gff2stats, 44
cgat.tools.gff2table, 64
cgat.tools.gff32gtf, 156
cgat.tools.gtf2fasta, 55
cgat.tools.gtf2gff, 45
cgat.tools.gtf2gtf, 50
cgat.tools.gtf2tsv, 137
cgat.tools.gtfs2tsv, 139
cgat.tools.index2bed, 118
cgat.tools.index_fasta, 68
cgat.tools.medip_merge_intervals, 119
cgat.tools.metaphlan2table, 158
cgat.tools.rnaseq_junction_bam2bam, 141
cgat.tools.split_fasta, 160
cgat.tools.split_file, 86
cgat.tools.split_gff, 142
cgat.tools.table2table, 79
cgat.tools.transfac2transfac, 162
cgat.tools.vcf2vcf, 122
cgat.tools.vcfstats2db, 123
cgat.tools.wig2bed, 163
FastaIterator, 175
Fastq, 176
Genomics, 190
GFF3, 179
GTF, 179
Histogram, 211
Histogram2D, 214
IGV, 209
IndexedFasta, 184
IndexedGenome, 187
Intervals, 196
Iterators, 220
Masker, 210
MatrixTools, 218
Motifs, 198
RateEstimation, 235
RLE, 229
SequencePairProperties, 198
SequenceProperties, 199
SetTools, 220
Sra, 189
Stats, 214
SVGdraw, 229
tools.bam_vs_gtf, 126
Tree, 222
TreeTools, 224
Variants, 205
VCF, 189
WrapperCodeML, 206
Motifs
    module, 198
move() (*SVGdraw.pathdata method*), 230

N

name (*Bed.Bed attribute*), 166
Newick2Nexus() (*in module TreeTools*), 224
Newick2Tree() (*in module TreeTools*), 224
Nexus2Newick() (*in module TreeTools*), 224
Nop() (*in module Tree*), 222
normalize() (*in module Histogram*), 213

O

Overlap() (*in module GTF*), 181

P

PairedTTest (*class in Stats*), 216
parse_region_string() (*in module Genomics*), 191
parseCoordinates() (*in module IndexedFasta*), 187
ParseFasta2Hash() (*in module Genomics*), 196
parseFrequencies() (*WrapperCodeML.BaseML method*), 208
parseGrids() (*WrapperCodeML.CodeMLSites method*), 208
parseInfo() (*GFF3.Entry method*), 179
parseInfo() (*GTF.Entry method*), 184
parseLog() (*WrapperCodeML.CodeML method*), 207
parseLog() (*WrapperCodeML.CodeMLPairwise method*), 208
parseOutput() (*WrapperCodeML.BaseML method*), 208
parseOutput() (*WrapperCodeML.CodeML method*), 208
parseOutput() (*WrapperCodeML.CodeMLPairwise method*), 208
parseOutput() (*WrapperCodeML.CodeMLSites method*), 208
parsePairs() (*WrapperCodeML.CodeMLPairwise method*), 208
parseRst() (*WrapperCodeML.CodeML method*), 207
parseSites() (*WrapperCodeML.CodeMLSites method*), 208
ParsingError, 168, 183, 206
path (*class in SVGdraw*), 232
pathdata (*class in SVGdraw*), 230
pattern (*class in SVGdraw*), 233
peek() (*in module Sra*), 189
point (*class in SVGdraw*), 232
polygon (*class in SVGdraw*), 232
polyline (*class in SVGdraw*), 232
pos() (*Variants.Variant property*), 205
pred() (*Stats.ROCResult property*), 217

prefetch() (*in module Sra*), 189
Print() (*in module Histogram*), 212
Print() (*in module Histogram2D*), 214
PrintAscii() (*in module Histogram*), 212
printPrettyAlignment() (*in module Genomics*), 196
process_remote_BAM() (*in module Sra*), 189
production pipeline, **238**
project pipeline, **238**
prune() (*in module Intervals*), 196
PruneTerminal() (*in module TreeTools*), 227
PruneTree() (*in module TreeTools*), 227
psl, **236**
pvalue() (*Stats.PairedTTest property*), 216
PysamIndexedFasta (*class in IndexedFasta*), 186
PYTHONPATH, 254

Q

qbezier() (*SVGdraw.pathdata method*), 230
quals (*Fastq.Record attribute*), 177
query, **237**
Quicksect (*class in IndexedGenome*), 188
quote() (*in module GTF*), 183

R

radialgradient (*class in SVGdraw*), 233
RateEstimation
    module, 235
rdf, **237**
read() (*GTF.Entry method*), 184
readAndIndex() (*in module Bed*), 167
readAndIndex() (*in module GTF*), 182
readAsIntervals() (*in module GTF*), 182
readContigSizes() (*in module Genomics*), 191
readFromFile() (*AGP.AGP method*), 164
readFromFile() (*in module GTF*), 182
ReadPeptideSequences() (*in module Genomics*), 196
ReconciliateByRio() (*in module TreeTools*), 227
Record (*class in Fastq*), 177
rect (*class in SVGdraw*), 231
reference() (*Variants.Variant property*), 205
regex2iupac() (*in module Motifs*), 198
relabel() (*Tree.Tree method*), 223
relbezier() (*SVGdraw.pathdata method*), 230
relellarc() (*SVGdraw.pathdata method*), 231
relhline() (*SVGdraw.pathdata method*), 230
relline() (*SVGdraw.pathdata method*), 230
relmove() (*SVGdraw.pathdata method*), 230
relqbezier() (*SVGdraw.pathdata method*), 231
relsmbezier() (*SVGdraw.pathdata method*), 230
relsmqbezier() (*SVGdraw.pathdata method*), 231
relvline() (*SVGdraw.pathdata method*), 230
RemoveFrameShiftsFromAlignment() (*in module Genomics*), 193
RemoveIntervalsContained() (*in module Intervals*), 197
RemoveIntervalsSpanning() (*in module Intervals*), 198
replicate, **237**
Reroot() (*in module TreeTools*), 227
rescaleBranchLengths() (*Tree.Tree method*), 223
resolveAmbiguousNA() (*in module Genomics*), 194
resolveReverseAmbiguousNA() (*in module Genomics*), 194
Result (*class in Stats*), 214
reverse_complement() (*in module Genomics*), 191
rfnr() (*Stats.ROCResult property*), 217
RLE
    module, 229
ROCResult (*class in Stats*), 217
root_at_node() (*Tree.Tree method*), 222
root_balanced() (*Tree.Tree method*), 223
root_midpoint() (*Tree.Tree method*), 223
rtpr() (*Stats.ROCResult property*), 217
run() (*WrapperCodeML.Evolver method*), 209
runEvolver() (*in module WrapperCodeML*), 209

S

sam, **237**
sample() (*in module Iterators*), 220
savitzky_golay() (*in module Stats*), 218
Scale() (*in module Histogram*), 212
score (*Bed.Bed attribute*), 166
score (*GTF.Entry attribute*), 183
script (*class in SVGdraw*), 234
seq (*Fastq.Record attribute*), 177
sequence (*FastaIterator.FastaRecord attribute*), 176
SequencePairInfo (*class in Genomics*), 195
SequencePairInfoCodons (*class in Genomics*), 195
SequencePairProperties
    module, 198
SequencePairPropertiesBaseML (*class in SequencePairProperties*), 198
SequencePairPropertiesCountsCodons (*class in SequencePairProperties*), 199
SequencePairPropertiesCountsNa (*class in SequencePairProperties*), 199
SequencePairPropertiesDistance (*class in SequencePairProperties*), 198
SequencePairPropertiesPID (*class in SequencePairProperties*), 199
SequenceProperties
    module, 199
SequenceProperties (*class in SequenceProperties*), 200

SequencePropertiesAA (*class in SequenceProperties*), 202
SequencePropertiesAminoAcids (*class in SequenceProperties*), 203
SequencePropertiesBias (*class in SequenceProperties*), 203
SequencePropertiesCodons (*class in SequenceProperties*), 203
SequencePropertiesCodonTranslator (*class in SequenceProperties*), 203
SequencePropertiesCodonUsage (*class in SequenceProperties*), 203
SequencePropertiesCounts (*class in SequenceProperties*), 204
SequencePropertiesCpg (*class in SequenceProperties*), 201
SequencePropertiesDegeneracy (*class in SequenceProperties*), 201
SequencePropertiesDN (*class in SequenceProperties*), 201
SequencePropertiesEntropy (*class in SequenceProperties*), 204
SequencePropertiesGaps (*class in SequenceProperties*), 201
SequencePropertiesHid (*class in SequenceProperties*), 200
SequencePropertiesLength (*class in SequenceProperties*), 200
SequencePropertiesNA (*class in SequenceProperties*), 200
SequencePropertiesSequence (*class in SequenceProperties*), 200
set (*class in SVGdraw*), 235
setConverter() (*IndexedFasta.cgatIndexedFasta method*), 186
setDefaultCaseList() (*CBioPortal.CBioPortal method*), 175
setDefaultStudy() (*CBioPortal.CBioPortal method*), 175
setFormat() (*Stats.DistributionalParameters method*), 215
setFrequencies() (*in module RateEstimation*), 236
setName() (*in module Bed*), 167
SetOptions() (*WrapperCodeML.BaseML method*), 208
SetOptions() (*WrapperCodeML.CodeML method*), 207
SetTools
    module, 220
setTranslator() (*IndexedFasta.cgatIndexedFasta method*), 186
setTree() (*WrapperCodeML.Evolver method*), 209
setUniformFrequencies() (*WrapperCodeML.Evolver method*), 208
setUniformFrequencies() (*WrapperCodeML.EvolverBaseml method*), 209
shift() (*in module cgat.tools.bed2bed*), 28
ShortenIntervalsOverlap() (*in module Intervals*), 198
Simple (*class in IndexedGenome*), 188
smbezier() (*SVGdraw.pathdata method*), 230
SmoothWrap() (*in module Histogram*), 212
smqbezier() (*SVGdraw.pathdata method*), 231
sort() (*IGV.IGV method*), 210
SortPerContig() (*in module GTF*), 182
source (*GTF.Entry attribute*), 183
spannedtext (*class in SVGdraw*), 231
Species2Genes() (*in module TreeTools*), 225
sphinxreport, **237**
splitFasta() (*in module IndexedFasta*), 187
Sra
    module, 189
sra, **237**
start (*Bed.Bed attribute*), 165
start (*GTF.Entry attribute*), 183
startIGV() (*in module IGV*), 209
statistic() (*Stats.PairedTTest property*), 216
Stats
    module, 214
stderr, **238**
stdin, **238**
stdout, **238**
stop (*class in SVGdraw*), 233
strand (*Bed.Bed attribute*), 166
strand (*GTF.Entry attribute*), 183
String2Alignment() (*in module Genomics*), 192
String2Location() (*in module Genomics*), 191
style (*class in SVGdraw*), 233
submit host, **237**
Summary (*class in Stats*), 215
svg, **237**
svg (*class in SVGdraw*), 235
SVGdraw
    module, 229
SVGelement (*class in SVGdraw*), 231
switch (*class in SVGdraw*), 234
switchTargetStrand() (*Blat.Match method*), 169
symbol (*class in SVGdraw*), 234

T

target, **237**
task, **237**
test directory, **237**
text (*class in SVGdraw*), 232
textpath (*class in SVGdraw*), 232
thickEnd (*Bed.Bed attribute*), 166
thickStart (*Bed.Bed attribute*), 166
title (*class in SVGdraw*), 233

title (*FastaIterator.FastaRecord attribute*), 176
tn() (*Stats.ROCResult property*), 217
tnr() (*Stats.ROCResult property*), 217
to_string() (*Tree.Tree method*), 222
toDot() (*in module GTF*), 183
toIntervals() (*Bed.Bed method*), 166
toIntronIntervals() (*in module GTF*), 182
tools.bam_vs_gtf
    module, 126
toPhred() (*Fastq.Record method*), 177
toSequence() (*in module GTF*), 182
tp() (*Stats.ROCResult property*), 217
tpr() (*Stats.ROCResult property*), 217
track, **237**
Track (*class in Bed*), 167
track_iterator() (*in module GTF*), 180
Transcript2GeneTree() (*in module TreeTools*), 226
transcript_id (*GTF.Entry attribute*), 183
transcript_iterator() (*in module GTF*), 180
translate() (*in module Genomics*), 194
TranslateDNA2Protein() (*in module Genomics*), 194
Translator (*class in IndexedFasta*), 185
TranslatorBytes (*class in IndexedFasta*), 185
TranslatorPhred (*class in IndexedFasta*), 185
TranslatorRange200 (*class in IndexedFasta*), 185
TranslatorSolexa (*class in IndexedFasta*), 185
traverseGraph() (*in module TreeTools*), 229
Tree
    module, 222
Tree (*class in Tree*), 222
Tree2Graph() (*in module TreeTools*), 227
Tree2Newick() (*in module TreeTools*), 224
TreeDFS() (*in module TreeTools*), 226
TreeTools
    module, 224
tref (*class in SVGdraw*), 231
trim() (*Fastq.Record method*), 177
trim5() (*Fastq.Record method*), 177
truncate() (*in module Intervals*), 197
truncate() (*Tree.Tree method*), 223
tspan (*class in SVGdraw*), 231
tss, **237**
tsv, **237**

U

unionIntersectionMatrix() (*in module SetTools*), 221
Unroot() (*in module TreeTools*), 227
updateNexus() (*in module Tree*), 222
updateProperties() (*SequenceProperties.SequencePropertiesDegeneracy method*), 202
updateProperties() (*Stats.DistributionalParameters method*), 215
updateVariants() (*in module Variants*), 205
upper() (*AString.AString method*), 190
UsageError, 206
use (*class in SVGdraw*), 234

V

value() (*Stats.ROCResult property*), 217
Variant (*class in Variants*), 205
Variants
    module, 205
VCF
    module, 189
vcf, **236**
VCFEntry (*class in VCF*), 190
VCFFile (*class in VCF*), 190
verify() (*in module IndexedFasta*), 187
view (*class in SVGdraw*), 234
vline() (*SVGdraw.pathdata method*), 230

W

wiggle, **236**
WrapperCodeML
    module, 206
Write() (*in module Histogram*), 212
WriteAlignment() (*WrapperCodeML.CodeML method*), 207
writeControlFile() (*WrapperCodeML.CodeML method*), 207
writeControlFile() (*WrapperCodeML.Evolver method*), 208
writeControlFile() (*WrapperCodeML.EvolverBaseml method*), 209
writeFragments() (*in module IndexedFasta*), 185
WriteNexus() (*in module TreeTools*), 224
writeSets() (*in module SetTools*), 221
writeToFile() (*Tree.Tree method*), 223
WriteTree() (*WrapperCodeML.CodeML method*), 207

X

xuniqueCombinations() (*in module SetTools*), 221

Y

yaml, **236**