## CONTENTS

1 **Segtools 1.1 documentation**  
   1.1 Installation ........................................ 3  
   1.2 Segmentations versus annotations .................. 4  
   1.3 Input ............................................... 4  
   1.4 Usage ................................................. 4  
   1.5 Commands ............................................ 7  
   1.6 Mnemonics .......................................... 18  
   1.7 Support .............................................. 19  

2 **Indices and tables**  

Index ...................................................... 23
Contents:
Segtools is a Python package for analyzing genomic segmentations. The software efficiently calculates a variety of summary statistics and produces corresponding publication quality visualizations. The overall goal of Segtools is to provide a bird’s-eye view of complex genomic data sets, allowing researchers to easily generate and confirm hypotheses.

For an overview of the toolkit, please see the manuscript:


Orion <stasis at uw dot edu> can send you the latest copy of the manuscript. Please cite the manuscript if you use Segtools.

**Note:** For questions, comments, or troubleshooting, please refer to the support section.

**1.1 Installation**

Segtools requires the following prerequisites:

- Python 2.5.1-2.7, with Rpy2 2.1.3+ and Genomedata packages
- Zlib
- Numpy 1.3+
- R 2.10.0+, with latticeExtra and reshape packages

Once these prerequisites are properly installed, Segtools can be installed with:

easy_install segtools
To help install these prerequisites and Segtools on any Linux platform, we developed a simple, interactive script. To download and run the script:

```bash
wget http://noble.gs.washington.edu/proj/segtools/install.py
python install.py
```

We are constantly trying to improve the installation script, so please let us know if you run into any trouble using it.

### 1.2 Segments versus annotations

A segmentation is a division of a genome (or part of a genome) into non-overlapping segments, each of which is assigned one of a fixed set of labels. Ideally, segments that share a common label are somehow similar to one another, and vice versa. Segtools helps you identify the similarities and differences between these labels to help you understand the semantics of your segmentation.

Some of the Segtools commands take annotations as input, rather than segmentations. The only difference between an annotation and a segmentation is that an annotation may contain overlapping segments (i.e., a single base can be assigned more than one label). Hence, any tool that accepts an annotation as input will also accept a segmentation, but the converse is not true.

Note that neither segmentations nor annotations need to be exhaustive, in the sense that the complete genome is labeled.

### 1.3 Input

Segmentations should be in BED format or GFF format, with one line for each segment and the name field used to specify the segment label. Segments must be non-overlapping, and can span all, part, or multiple parts of a genome. Genomic regions not spanned by any segment are ignored, so it can sometimes be useful to have a “background” label with segments that span all regions not covered by another segmentation. Adding these labels can be automated with `segtools-flatten`. For best results, the number of unique segment labels should be between 2 and around 40. For segmentations, Segtools uses fields 1–4 of a BED file or fields 1, 3–5 of a GFF file.

Annotations (i.e., segmentations in which overlapping segments are allowed) may be specified in BED format, GFF format, or GTF format.

If you want to change the order in which labels appear or if you want to change the text displayed in plots, a mnemonic file can be created. Segtools commands can then be re-run with the `--replot` flag and the `--mnemonic-file=<FILE>` option to regenerate the plots without redoing the computation.

### 1.4 Usage

The basic workflow for using key Segtools commands is shown below. Segmentations can also be created from other segmentations, annotations, or peak-calls using `segtools-flatten`. 
Segtools commands can be run through their command-line or Python interfaces.

### 1.4.1 Command-line interface

Commands that analyze a single segmentation:

- **segtools-length-distribution**: Analyzes the distribution of segment lengths and their coverage of the genome for each segment label.
- **segtools-nucleotide-frequency**: Analyzes the frequencies of nucleotides and dinucleotides in segments of each label.
- **segtools-transition**: Analyzes the transitions between segment labels and the structure of their interaction.

Commands that compare a segmentation to an annotation:

- **segtools-aggregation**: Analyzes the relative occurrence of each segment label around the provided genomic features.
- **segtools-feature-distance**: Reports the distance from each segment to the nearest feature in each of a list of feature files.
- **segtools-overlap**: Analyzes the frequency with which each segment label overlaps features of various types.
- **segtools-signal-distribution**: Analyzes the distribution of genomic signal tracks for each segment label.

Segtools utilities:

- **segtools-compare**: Measure base-wise edit distance between two segmentations.
- **segtools-flatten**: General tool for flattening overlapping segments, but flattens them into segments defined by the set of segment labels that overlap the region.
- **segtools-html-report**: Combines the output of the other commands and generates an html report for easier viewing.
- **segtools-preprocess**: Preprocess segmentation and annotation files into a binary format that can be quickly re-read in future calls to Segtools commands.
• *segtools-relabel*: Generate a new segmentation by relabeling the segments in one segmentation according to a mnemonic file.

All the above commands respond to `-h` and `--help` by displaying usage information and options.

Where relevant, commands accept *mnemonic files* through the `--mnemonic-file` option.

Each core command generates:

- tab-delimited (*tab*) data files
- image files (in *png* and *pdf* format and in normal, thumbnail, and slide layouts), and
- partial HTML (*div*) files, which are used by *segtools-html-report*.

**Common options**

The following options are supported by all (or most) Segtools commands:

- **-clobber**
  - Overwrite existing output files if there is a conflict.

- **-help, -h**
  - Display usage information and options.

- **-noplot**
  - Perform computation and output tab files, but do not generate plots.

- **-mnemonic-file <file>, -m <file>**
  - Specify a mnemonic file to control the label display and ordering. See *mnemonic file details*.

- **-outdir <dir>, -o <dir>**
  - Specify the directory where output files should be placed (will be be generated if it does not exist).

- **-quick**
  - Output results after running command on only one chromosome (which chromosome is unspecified). This can be useful for testing.

- **-quiet, -q**
  - Don’t print diagnostic messages and status updates.

- **-replot**
  - Load tab file data generated from a previous run of this program and recreate plots instead of recomputing tab file data. Tab files are expected to be in the default or specified output directory (with `-outdir`).

- **-version**
  - Print the current program version.

### 1.4.2 Python interface

Segtools commands can be run directly from Python by importing the corresponding module and running its `main()` method with the same arguments you would specify on the command line. For instance, you could run `segtools-length-distribution -opt ARG` from Python with the following:

```python
>>> from segtools import length_distribution
>>> length_distribution.main(["-opt", "ARG"])
```

The commands and corresponding modules are as follows:
<table>
<thead>
<tr>
<th>Command (segtools-...)</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>aggregation</td>
<td>aggregation</td>
</tr>
<tr>
<td>compare</td>
<td>compare</td>
</tr>
<tr>
<td>feature-distance</td>
<td>feature_distance</td>
</tr>
<tr>
<td>flatten</td>
<td>flatten</td>
</tr>
<tr>
<td>html-report</td>
<td>html</td>
</tr>
<tr>
<td>length-distribution</td>
<td>length_distribution</td>
</tr>
<tr>
<td>nucleotide-frequency</td>
<td>nucleotide_frequency</td>
</tr>
<tr>
<td>overlap</td>
<td>overlap</td>
</tr>
<tr>
<td>preprocess</td>
<td>preprocess</td>
</tr>
<tr>
<td>relabel</td>
<td>relabel</td>
</tr>
<tr>
<td>signal-distribution</td>
<td>signal_distribution</td>
</tr>
<tr>
<td>transition</td>
<td>transition</td>
</tr>
</tbody>
</table>

1.5 Commands

1.5.1 Commands that analyze a single segmentation

segtools-length-distribution

This command summarizes the distribution of segment lengths, by label. It generates a violin plot (a box plot, but instead of a box, it is a smoothed density curve), a simple bar chart that describes the overall label composition of the annotations file, and a table listing the number of segments of each label, the mean and median segment lengths, and the number of bases covered by each label.

Usage: segtools-length-distribution [OPTIONS] ANNOTATIONS

Options:

- **--version**
  show program’s version number and exit

- **-h, --help**
  show this help message and exit

Flags:

- **--clobber**
  Overwrite existing output files if the specified directory already exists.

- **-q, --quiet**
  Do not print diagnostic messages.

- **--replot**
  Load data from output tab files and regenerate plots instead of recomputing data

- **--noplot**
  Do not generate plots

- **--no-segments**
  Do not show total segments covered by labels in size plot

- **--no-bases**
  Do not show total bases covered by labels in size plot

Output:

- **-m FILE, --mnemonic-file=FILE**
  If specified, labels will be shown using mnemonics found in FILE.

- **-o DIR, --outdir=DIR**
  File output directory (will be created if it does not exist) [default: length_distribution]

**Note:** This command requires only an annotations file as a parameter and performs minimal computation. As such, it is a useful test to make sure Segtools works on your system.
Note: Since the violin plot is based upon a density distribution, the lengths of all the segments in the segmentation or annotations is saved in a tab file to allow this plot to be regenerated solely from R. Unfortunately, for large annotations, this tab file can get very large (hundreds of megabytes). We hope to revise this by saving instead a histogram-like summary of the segment lengths instead of a separate length for each segment.

segtools-nucleotide-frequency

This command generates a heatmap of the normalized dinucleotide frequencies found across segments of each label, as well as table of such nucleotide and dinucleotide frequencies.

As input, the command takes a annotations file and a Genomedata archive for the genome the annotations cover. The Genomedata archive is used solely for the nucleotide sequence.

Usage: segtools-nucleotide-frequency [OPTIONS] ANNOTATIONS GENOMEDATAFILE

Options:

- --version
  show program’s version number and exit

- -h, --help
  show this help message and exit

Flags:

- --clobber
  Overwrite existing output files if the specified directory already exists.

- -q, --quiet
  Do not print diagnostic messages.

- --quick
  Compute values only for one chromosome.

- --replot
  Load data from output tab files and regenerate plots instead of recomputing data

- --noplot
  Do not generate plots

Output:

- -m FILE, --mnemonic-file=FILE
  If specified, labels will be shown using mnemonics found in FILE.

- -o DIR, --outdir=DIR
  File output directory (will be created if it does not exist) [default: nucleotide_frequency]

segtools-transition

This command takes a segmentation and computes the frequencies all all possible transitions between segment labels. If a segment with label A is immediately followed by a segment with label B, then this is counted as one A->B transition. This command is thus most useful for segmentations that are a partition of large regions or the whole genome. If your segmentation consists of a collection of peak calls or regions of interest, then it is unlikely that there are many pairs of directly adjacent segments, and the results will be meaningless.

As output, this command generates a heatmap of transition frequencies as well as a graph interpretation of this heatmap. In many cases, there will be at least one transition between every pair of segment labels, making the transition graph fully connected. To make the graph more interpretable, the transition frequencies can be thresholded by value (--prob-threshold) or quantile (--quantile-threshold) before drawing the edges.

Usage: segtools-transition [OPTIONS] SEGMENTATION
Options:

--version show program’s version number and exit
-h, --help show this help message and exit

Flags:

--clobber Overwrite existing output files if the specified directory already exists.
-q, --quiet Do not print diagnostic messages.
--replot Regenerate graphs/plots from tab files
--noplot Do not generate transition plots
--nograph Do not generate transition graph

Output:

-m FILE, --mnemonic-file=FILE If specified, labels will be shown using mnemonics found in FILE.
-o DIR, --outdir=DIR File output directory (will be created if it does not exist) [default: transition]

Transition frequency plot options:

--dd, --dendrogram include dendrogram along edge of levelplot [default: False]

Transition graph options:

-P VAL, --prob-threshold=VAL ignore all transitions with probabilities below this absolute threshold [default: 0.15]
-Q VAL, --quantile-threshold=VAL ignore transitions with probabilities below this probability quantile [default: 0.0]

Non-segmentation files:

-g, --gmtk The SEGMENTATION argument will instead be treated as a GMTK parameter file. If a mnemonic file is not specified, one will be created and used.

1.5.2 Commands that compare a segmentation to an annotation

segtools-aggregation

This command looks at the aggregate occurrence of segment labels around and within annotated features. A typical example of this would be to look at the relative occurrences of segment labels around transcription start sites (TSSs). You would do this with something like:

segtools-aggregation --normalize segmentation.bed tss.gff

If you have two different classes of TSSs that you were interested in (say, expressed and unexpressed), you can use the third column of the GFF file as a grouping variable and then specify the --groups flag.

By default, the y-axis of the aggregation plot is the number of segments of a given label that overlap a region. This is useful in some applications, but more often you are interested in the enrichment or depletion of various labels relative to what you might expect by chance. This is especially true if the segments in one label are significantly longer than those in another label. In these cases, the --normalize flag should be used.
Usage: segtools-aggregation [OPTIONS] SEGMENTATION ANNOTATIONS

Plot the enrichment of the SEGMENTATION labels relative to the position of features in ANNOTATIONS. Features can be grouped by the ‘name’/’feature’ column by supplying –groups.

Options:

--version show program’s version number and exit
-h, --help show this help message and exit

Output options:

-m FILE, --mnemonic-file=FILE If specified, labels will be shown using mnemonics found in FILE
-o DIR, --outdir=DIR File output directory (will be created if it does not exist) [default: feature_aggregation]

Flags:

--clobber Overwrite existing output files if the specified directory already exists.
-q, --quiet Do not print diagnostic messages.
--quick Compute values only for one chromosome.
--replot Load data from output tab files and regenerate plots instead of recomputing data
--noplot Do not generate plots
--groups Separate data into different groups based upon ANNOTATIONS’s ‘name’/’feature’ field if –mode=region or –mode=point. Does nothing if –mode=gene.
--normalize Plot the relative frequency of SEGMENTATION labels, normalized by the number of segments in that group instead of the raw counts (normalize over SEGMENTATION labels)

Main aggregation options:

--mode=MODE one of: point, region, gene. [default: point]
-f N, --flank-bases=N Aggregate this many base pairs off each end of feature or gene [default: 500]
-r N, --region-samples=N If –mode=region, aggregate over each internal feature by taking this many evenly-spaced samples. Warning: features with a length < N will be excluded from the results [default: 50]

Gene aggregation options:

-i N, --intron-samples=N Aggregate over each intron by taking this many evenly-spaced samples [default: 50]
-e N, --exon-samples=N Aggregate over each exon by taking this many evenly-spaced samples [default: 25]

Further description of selected options:

--mode <mode>
Specify the aggregation mode. The following options are available: point, region, and gene. The default is point.
point: This mode aggregates around point-like features such as TSSs, TESs, or single-base peak calls. This mode looks at where segments occur in the 5' and 3' flanking regions of each feature. If the feature annotations have strand specifications (column 7), then the aggregation is strand-corrected so that the 5' flank region is always upstream of the feature. The width (in base pairs) of these flanking regions can be set with the --flank-bases option (default 500 bp).

region: This mode aggregates around region-like features such as transcription factor binding sites, ChIP-seq peak calls, or promoter regions. This will be the appropriate mode for most annotations. This mode is similar to point, but with the addition of an internal region which is aggregated over as well. To account for regions of varying length, evenly-spaced samples are taken from the span of each feature. The number of these samples can be set with --region-samples. Features that span fewer bases than this sample number are skipped.

gene: This is a special mode for aggregating with respect to an idealized gene model. Rather than specifying a normal GFF file, the annotation file must be in GTF format and have features with names exon and CDS, as provided by exporting data from the UCSC Table Browser in GTF format. This mode is similar to region, but with many regions that correspond to idealized transcriptional and translational models of genes. For the transcriptional model, there are regions corresponding to initial, internal, and terminal exons and introns. For the translational model, there are initial, internal, and terminal 5' and 3' UTR regions, and initial and terminal CDSs. In the visualization output, each type of model is laid out in a logical progression so that genes are viewed in profile and gene-component-specific associations can be easily seen. Because introns and exons are typically different lengths, the --intron-samples and --exon-samples options allow you to specify the number of samples that are taken in these regions (like in region mode). Note: If there are multiple transcripts with the same gene ID, then the longest transcript is used.

-normalize
This option normalizes the y-axis of the aggregation plot, displaying enrichment and depletion instead of counts at each position. The enrichment of label $l$ at position $p$ is calculated with the following formula:

$$enrichment(l, p) = \log_2 \frac{f_{obs} + 1}{f_{rand} + 1}$$

where $f_{obs}$ is the observed overlap frequency and $f_{obs}$ is the overlap frequency expected at random, defined by:

$$f_{obs} = \frac{\text{count}(l, p)}{\sum_{labels} \text{count}(p)}$$

$$f_{rand} = \frac{\text{bases_in_label}(l)}{\sum_{labels} \text{bases_in_label}}$$

The enrichment is thus bounded by $[-1, 1]$, with 1 corresponding to extreme enrichment and -1 to extreme depletion.

-groups
Group the features by the value of the third column of the GFF or GTF file (the name field). This option is useful if you want to compare aggregation profiles with respect to multiple classes of features, such as TSSs split by expression level or cell type.

-segtools-feature-distance
This command takes a segmentation and one or more annotations files and prints the distance from each segment to the nearest annotation in each file. Results are printed in tab-delimited format on stdout:

<table>
<thead>
<tr>
<th>chrom</th>
<th>start</th>
<th>end</th>
<th>label</th>
<th>...</th>
</tr>
</thead>
</table>

where ... is a tab-delimited list of distances, one per annotation file.

This command can be used in conjunction with other command-line UNIX utilities to easily sort and filter segments by their distance from important genomics features. For example, given a segmentation from genomic insulator sites,
you could use this command to find the 100 insulators farthest from any transcription start site. The command can also be used to filter segments that overlap annotation sets by filtering for distances of 0.

Usage: segtools-feature-distance [OPTIONS] SEGMENTATION ANNOTATIONS

Given a SEGMENTATION and ANNOTATIONS file, 1) prints the distance of each segment to the nearest feature in the ANNOTATIONS file (zero if the two overlap) and 2) generates a histogram of these distances. Distance is the difference between the nearest bases of the segment and the annotation, so if there is one base pair between them, the distance is 2. Distances can be strand-corrected with respect to stranded annotations by specifying --stranded.

Options:

--version show program’s version number and exit
-h, --help show this help message and exit

Basic options:

--clobber Overwrite existing output files if the specified directory already exists.
-q, --quiet Do not print diagnostic messages.
-s, --stranded Strand correct annotations in the ANNOTATIONS file. If the annotations contains strand information, the sign of the distance value is used to portray the relationship between the segment and the nearest stranded annotation. A negative distance value indicates that the segment is nearest the 5’ end of the annotation, whereas a positive value indicates the segment is nearest the 3’ end of the annotation.

Print options:

-p, --print-nearest The name of the nearest annotation will be printed after each distance (with a space separating the two) for annotations from the ANNOTATIONS file. If multiple features are equally near (or overlap), it is undefined which will be printed
-a, --all-overlapping If multiple annotations in the ANNOTATIONS file overlap a segment, a separate line is printed for each overlapping segment-annotation pair. This is most interesting with --print-nearest. Otherwise, the first overlapping segment will be used for printing.

Plotting options:

--noplot Do not generate plots
--replot Load data from output tab files and regenerate plots instead of recomputing data
-n N, --n-bins=N Number of bins to use in histogram for distances greater than zero and less than or equal to N*W. Distances of 0 and greater than N*W are placed in additional bins. If --stranded, N more bins are included in the negative direction and a bin for distances less than -N*W.
-w W, --bin-width=W Number of bases in each bin. If --stranded, bins cover distances of (-Inf,-N*W), ..., [-W,0], [0,W], ..., (N*W,Inf). Otherwise, the bins cover distances of [0], (0,W], (W,2W], ..., (N*W,Inf).

Output options:

-m FILE, --mnemonic-file=FILE If specified, labels will be shown using mnemonics found in FILE
segtools Documentation, Release 1.1.7

-o DIR, --outdir=DIR  File output directory (will be created if it does not exist) [default: feature_distance]

segtools-overlap

segtools-overlap [OPTIONS] SEGMENTATION ANNOTATIONS

This command measures the base-wise or segment-wise overlap between segments in a segmentation and an annotations file. Segments are classified by their label and annotations can be classified with a group, so the primary output is a confusion matrix with each cell representing the amount of overlap between segments in one label with annotations in one group. Further, the ability of each segment label to predict each annotation group is measured and summarized in a precision-recall plot or a confusion matrix heat map.

Usage: segtools-overlap [OPTIONS] SEGMENTATION ANNOTATIONS

SEGMENTATION and ANNOTATIONS files should be in BED, GFF, or GTF format (grouped on ‘name’/’feature’ columns). Results summarize the overlap of SEGMENTATION groups with ANNOTATIONS groups. The symmetric analysis can be performed by rerunning the program with the input file arguments swapped (and a different output directory). A rough specification can be found here: http://en-codewiki.ucsc.edu/EncodeDCC/index.php/Overlap_analysis_tool_specification

Options:

--version  show program’s version number and exit
-h, --help  show this help message and exit

Flags:

--clobber  Overwrite existing output files if the specified directory already exists.
-q, --quiet  Do not print diagnostic messages.
--quick  Compute values only for one chromosome.
--replot  Load data from output tab files and regenerate plots instead of recomputing data
--noplot  Do not generate plots
--cluster  Cluster rows and columns in heat map plot
-p, --print-segments  For each group in the SEGMENTATION, a separate output file will be created that contains a list of all the segments that the group was found to overlap with. Output files are named overlap.segments.X.txt, where X is the name of the SEGMENTATION group.
--max-contrast  Saturate color range instead of having it go from 0 to 1

Parameters:

-b MODE, --by=MODE  One of: ['segments', 'bases'], which determines the definition of overlap. @segments: The value associated with two features overlapping will be 1 if they overlap, and 0 otherwise. @bases: The value associated with two features overlapping will be number of base pairs which they overlap. [default: bases]

--min-overlap=N  The minimum number of base pairs that two features must overlap for them to be classified as overlapping. This integer can be either positive (features overlap only if they share at least this many bases) or negative (features overlap if there are no more than this many bases

1.5. Commands
between them). Both a negative min- overlap and –by=bases cannot be specified together. [default: 1]

Files:

- **-m FILE, --mnemonic-file=FILE** If specified, SEGMENTATION groups will be shown using mnemonics found in FILE.

--**feature-mnemonic-file=FILE** If specified, ANNOTATIONS groups will be shown using mnemonics found in FILE.

- **-o DIR, --outdir=DIR** File output directory (will be created if it does not exist) [default: overlap]

Additional description of selected options:

- **-min-overlap <n>**
  This specifies the minimum number of bases that a segment and an annotation must overlap for that overlap to be counted. This number can be positive or negative, with <n> = 1 indicating that the segment and annotation must overlap by at least one base, a <n> = 0 indicating that they can be directly adjacent, and <n> = -1 indicating that there can be one base separation for them to still count as overlapping.

Note: The precision-recall plot is only generated for base-wise overlap.

**segtools-signal-distribution**

This command takes a segmentation and a Genomedata archive and summarizes the distribution of values for each Genomedata track that fall within segments of each label. Essentially, it generates a histogram for each label-track pair, where the values being measured are the values for that track found in segments of that label. Currently, Segtools no longer generates this matrix of histograms, but instead generates a heatmap of mean and standard deviation values.

Note: Mean and standard deviation values are approximated from a histogram (binned) representation of the data. The effect should be minimal, but it is important to keep this in mind as a potential source of error.

**Parallelization:**

Depending upon the segmentation and the Genomedata archive, this command can take a very long time to run. To help speed it up, you can parallelize the command by chromosome. To do this, you would first submit a job for each chromosome that doesn’t plot anything, then merge the results to generate the final output. Here is sample pseudocode:

```bash
indirs = []
for <chrom> in <chroms>
    outdir = sub_<chrom>
    submit_job --name=<run_id> segtools-signal-distribution --noplot --chrom=<chrom> --outdir=<outdir>
    indirs.add("--indir=" + <outdir>)

submit_job --hold_on=<run_id> segtools-signal-distribution --indirs=<indirs>
```

Usage: segtools-signal-distribution [OPTIONS] SEGMENTATION GENOMEDATAFILE

Options:

- **--version** show program’s version number and exit
- **-h, --help** show this help message and exit

Flags:
**-clobber**
Overwrite existing output files if the specified directory already exists.

**-q, --quiet**
Do not print diagnostic messages.

**--quick**
Compute values only for one chromosome.

**--noplot**
Do not generate plots

**--replot**
Load data from output tab files and regenerate plots instead of recomputing data

**--create-mnemonics**
If mnemonics are not specified, they will be created and used for plotting

**--ecdf**
Plot empirical cumulative density inside each panel instead of a normal histogram (turns off log-y)

**--calc-ranges**
Calculate ranges for distribution plots from segmentation data (slower) instead of using whole genome data (default).

**Histogram options:**

- **-n N, --nbins=N**
Number of bins for signal distribution [default: 100]

- **-c CHROM, --chrom=CHROM**
Only perform the analysis on data in CHROM, where CHROM is a chromosome name such as chr21 or chrX (option can be used multiple times to allow multiple chromosomes)

**I/O options:**

- **-m FILE, --mnemonic-file=FILE**
If specified, labels will be shown using mnemonics found in FILE

- **--order-tracks=FILE**
If specified, tracks will be displayed in the order in FILE. FILE must be a permutation of all the printed tracks, one per line, exact matches only.

- **--order-labels=FILE**
If specified, label will be displayed in the order in FILE. FILE must be a permutation of all the labels (after substituting with mnemonics, if specified), one per line, exact matches only.

- **-i DIR, --indir=DIR**
Load data from this directory This directory should be the output directory of a previous run of this module. This option can be specified multiple times, making it useful for parallelizing this command, since multiple results can be merged together to generate the final output.

- **-o DIR, --outdir=DIR**
File output directory (will be created if it does not exist) [default: signal_distribution]

**Additional description of selected options:**

- **-chrom <chrom>, -c <chrom>**
Restrict the analysis to the single chromosome specified. This option is useful in parallelizing this command. <chrom> must exactly match a chromosome in the Genomedata archive (genome[<chrom>] must be valid).

- **-create-mnemonics**
Use the hierarchically-clustered heatmap of mean values to generate mnemonics for the segmentation labels. The mnemonic labels are of the form: X.Y, where X is the group number and Y is the index in that group.

- **-indir <dir>, -i <dir>**
Load data from the output directory of a previous run of this command. This option can be specified multiple times, making it useful for parallelizing this command, since multiple results can be merged together to generate the final output.
1.5.3 Segtools utilities

segtools-compare

This command compares two segmentations by a specified metric. Currently, the only supported metric is \texttt{--edit-distance}.

Usage: segtools-compare [OPTIONS] SEGMENTATION SEGMENTATION

Compare two segmentations

Options:

- \texttt{--version} show program’s version number and exit
- \texttt{-h, --help} show this help message and exit
- \texttt{-d, --edit-distance} Measure the base-wise edit distance between the two specified segmentations
- \texttt{--quick} Output results after only one chromosome
- \texttt{-q, --quiet} Don’t print diagnostic messages

segtools-flatten

This command takes multiple segmentations and combinatorially flattens them into one. Thus, there is a segment boundary in the new segmentation for every segment boundary in any of the input segmentations. The label for each new segment is a concatenation of the labels for segments that overlap this segment. We hope to expand this to support annotations files in the near future.

For example, given two files of regions of high transcription factor binding, one with peak calls and one with a lower threshold, you could create a single segmentation from the two files with:

\texttt{segtools-flatten peaks.gff regions.bed.gz}

The new segmentation would have one segment label for bases that are covered by regions.bed.gz but not peaks.gff, one for bases covered by both files, and one for bases covered by only peaks.gff (if there are any). The command prints the new segmentation in BED format to standard output, by default, but \texttt{--mnemonic-file} and \texttt{--outfile} can be specified to create a segmentation file with a corresponding mnemonic file that can be used in further Segtools analyses.

Usage: segtools-flatten [OPTIONS] SEGMENTATION...

Combine segments from all SEGMENTATION files, labeling with unique labels for each combination of overlapping that occurs. Outputs a bed file on stdout (-o to change), and generates a file in the current directory that describes the generated labels (use -m to change).

Options:

- \texttt{--version} show program’s version number and exit
- \texttt{-h, --help} show this help message and exit
- \texttt{-q, --quiet} Do not print diagnostic messages.
- \texttt{-m FILE, --mnemonic-file=FILE} Save mapping information to FILE instead of flat.mnemonics (default). This file complies with the mnemonic file format.
- \texttt{-o FILE, --outfile=FILE} Save flattened bed file to FILE instead of printing to stdout (default)
-f F, --filter=F  Don’t output new segment labels (and corresponding segments) that span less than F*N bases, where N is the number of bases covered by the new segmentation. This can be used to remove extremely uncommon labels (e.g. F = 0.01) that are the more likely to be spurious. Filtering is off by default.

**segtools-html-report**

This command is intended to be run after other Segtools commands. Starting in the current working directory (or directory provided with --results-dir), the command finds files produced by the other Segtools commands (files matching *//*.div) and compiles the results into an HTML report for review.

The SEGMENTATION argument and --mnemonic-file option should be the same as used to run the other Segtools commands.

Usage: segtools-html-report [OPTIONS] SEGMENTATION

Options:

- --version  show program’s version number and exit
- -h, --help  show this help message and exit

Flags:

- --clobber  Overwrite existing output files if the specified directory already exists.
- -q, --quiet  Do not print diagnostic messages.

Linking:

- -m FILE, --mnemonic-file=FILE  If specified, this mnemonic mapping will be included in the report (this should be the same mnemonic file used by the individual modules).
- -L FILE, --layered-bed=FILE  If specified, this layered BED file will be linked into the HTML document (assumed to be the same data as in SEGMENTATION)
- -B FILE, --big-bed=FILE  If specified, this bigBed file will be linked into the HTML document and a UCSC genome browser link will be generated for it (assumed to be the same data as in SEGMENTATION)

Output:

- --results-dir=DIR  This should be the directory containing all the module output directories (ls should return things like “length_distribution/”, etc) [default: .]
- -o FILE, --outfile=FILE  HTML report file (must be in current directory or the links will break [default: index.html]

**segtools-preprocess**

This command takes a segmentation or annotations file (INFILE) and generates a binary, preprocessed file (INFILE.pkl) that can be quickly loaded in subsequent calls to Segtools commands. This is especially useful if you want to run many Segtools commands on one segmentation. If you don’t preprocess the segmentation, each Segtools command parses the segmentation file independently. If the segmentation is large, this can add an hour or more to the
runtime of each Segtools command. Preprocessing cuts this load time to just a few seconds. See --help for more details.

Usage: segtools-preprocess [OPTIONS] INFILE [OUTFILE]

If Segtools is taking too long to parse your large segmentation or annotation file (specify --annotation), this tool allows you to pre-process that file only once and have it load much faster in the future. INFILE will be parsed to create a special binary file with a name of the form: “INFILE.pkl” or “OUTFILE.pkl”. Then, you can substitute this new file for the corresponding SEGMENTATION or ANNOTATIONS argument in future Segtools calls and Segtools will parse it in much faster (but be sure to keep the “.pkl” extension intact)!

Options:

- --version show program’s version number and exit
- -h, --help show this help message and exit
- --clobber Overwrite existing output files without warning
- -q, --quiet Do not print diagnostic messages.
- --annotation Read INFILE as a set of annotations, rather than as a segmentation (default).

segtools-relabel

This command generates a new segmentation by remapping the labels in a segmentation according to a mnemonic file. Multiple old labels can be mapped to the same new label. Immediately adjacent segments that have the same new label are merged, so that in the new segmentation, no two consecutive segments have the same label.

Usage: segtools-relabel [OPTIONS] SEGMENTATION MNEMONICFILE

Generates a new segmentation by first relabeling the segments in a given SEGMENTATION according to the label mappings in MNEMONICFILE and then merging overlapping segments with the same final label. Outputs the new segmentation in bed format to stdout (-o to change).

Options:

- --version show program’s version number and exit
- -h, --help show this help message and exit
- -q, --quiet Do not print diagnostic messages.
- -o FILE, --outfile=FILE Save relabeled bed file to FILE instead of printing to stdout (default)

1.6 Mnemonics

Mnemonic files are supported by most of the Segtools commands and provide a way to rename and reorder the displayed labels without repeating the full analysis. Mnemonic files must contain two or three tab-separated columns and must contain start with the following header (the description column is optional):

old{TAB}new[{TAB}description]

Renaming:

Each line of the mnemonic file specifies a mapping from the “old” name (the one appearing in the segmentation file) to the “new” name (the one to be displayed). Because the new name must fit into small spaces in plots (such as axis labels), this field should not be longer than a few characters (such as “I” for insulator). Longer descriptions can be specified in the description column.
Reordering:

The order of the lines in the mnemonic file determines the order the labels will be shown in plots.

Example:

If the segmentation file contains segments with labels of A, B, and C, but you want A to be displayed as A1, C to be displayed as A2, and the two of them to be next to each other in plots, then you should construct the following mnemonic file:

<table>
<thead>
<tr>
<th>old</th>
<th>new</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A1</td>
</tr>
<tr>
<td>C</td>
<td>A2</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
</tr>
</tbody>
</table>

Including the B line is not necessary, but it makes it easier to reorder the labels later (for instance, if you want B to come first). A description column could also have been included. This file should be saved as something like second_try.mnemonics and should be passed into Segtools commands with --mnemonic-file=/path/to/second_try.mnemonics.

If you had previously run Segtools commands on the segmentation before creating these mnemonics, you could speed up the plot corrections by using the commands’ --replot option (all other options and arguments should still be specified to ensure correctness).

1.7 Support

For support of Segtools, please write to the segtools-users@uw.edu mailing list, rather than writing the authors directly. Using the mailing list will get your question answered more quickly. The mailing list also allows us to pool knowledge and reduce the number of repeated inquiries. You can subscribe here:

https://mailman1.u.washington.edu/mailman/listinfo/segtools-users

If you want to report a bug or request a feature, please do so using our issue tracker:

http://code.google.com/p/segtools/issues

If you do not want to read discussions about other people’s use of Segtools, but would like to hear about new releases and other important information, please subscribe to segtools-announce@uw.edu by visiting this web page:

https://mailman1.u.washington.edu/mailman/listinfo/segtools-announce
INDICES AND TABLES

- genindex
- modindex
- search
INDEX

Symbols
- `–chrom <chrom>, -c <chrom>`
  - segtools-signal-distribution command line option, 15
- `–clobber`
  - command line option, 6
- `–create-mnemonics`
  - segtools-signal-distribution command line option, 15
- `–groups`
  - segtools-aggregation command line option, 11
- `–help, -h`
  - command line option, 6
- `–indir <dir>, -i <dir>`
  - segtools-signal-distribution command line option, 15
- `–min-overlap <n>`
  - segtools-overlap command line option, 14
- `–mnemonic-file <file>, -m <file>`
  - command line option, 6
- `–mode <mode>`
  - segtools-aggregation command line option, 10
- `–noplot`
  - command line option, 6
- `–normalize`
  - segtools-aggregation command line option, 11
- `–outdir <dir>, -o <dir>`
  - command line option, 6
- `–quick`
  - command line option, 6
- `–quiet, -q`
  - command line option, 6
- `–replot`
  - command line option, 6
- `–version`
  - command line option, 6

C
command line option
- `–clobber, 6`
- `–help, -h, 6`
- `–mnemonic-file <file>, -m <file>, 6`
- `–noplot, 6`
- `–outdir <dir>, -o <dir>, 6`
- `–quick, 6`
- `–quiet, -q, 6`
- `–replot, 6`
- `–version, 6`