CONTENTS

1 Segtools 1.1 documentation 3
   1.1 Installation ........................................ 3
   1.2 Basics .............................................. 4
   1.3 Input ................................................ 4
   1.4 Usage ................................................ 4
   1.5 Commands ............................................ 6
   1.6 Mnemonics .......................................... 11
   1.7 Support ............................................. 12

2 Indices and tables 13

Index 15
SEGTOOLS 1.1 DOCUMENTATION

Homepage  http://noble.gs.washington.edu/proj/segtools
Author  Orion Buske <stasis at uw dot edu>
Organization  University of Washington
Address  Department of Genome Sciences, PO Box 355065, Seattle, WA 98195-5065, United States of America
Copyright  2009-2010, Orion Buske; 2010 Michael M. Hoffman
Last updated  March 12, 2011

Segtools is a Python package designed to put genomic segmentations back in the context of the genome! Using R for graphics, Segtools provides a number of modules to analyze a segmentation in various ways and help you interpret its biological relevance.

For a broad overview, please see the manuscript:

Under revision.

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

A simple, interactive script has been created to install Segtools (and most dependencies) on any Linux platform. Installation is as simple as downloading and running this script! For instance:

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

Note:  The following prerequisites must be installed first:

- Python 2.5.1-2.7
- Zlib
1.2 Basics

A segmentation is typically a partition of a genome (or part of a genome) into non-overlapping segments, each of which is assigned one of a small set of labels. The idea is that segments that share a common label are somehow similar, and those that have different labels are somehow different. Segtools helps you identify the similarities and differences between these labels to help you understand your segmentation at a higher level.

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. This 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.

If you want to change the order in which labels appear or the text displayed in plots, a Mnemonics can be created. Segtools commands can be re-run with the --replot flag and the --mnemonic-file=<FILE> option to regenerate the plots without redoing the computation. Similarly, mnemonic files can be swapped or revised and new images created with relative ease.

Most Segtools commands look for patterns between segment labels in a segmentation and some known annotation. For such commands, the annotations are often specified in BED format or GFF format (although some commands require GTF or Genomedata formats).

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

Core commands:

- **segtools-aggregation**: Analyzes the relative occurrence of each segment label around the provided genomic features.
- **segtools-transition**: Analyzes the transitions between segment labels and the structure of their interaction.
- **segtools-length-distribution**: Analyzes the distribution of segment lengths and their coverage of the genome for each segment label.
- **segtools-signal-distribution**: Analyzes the distribution of genomic signal tracks for each segment label.
- **segtools-nucleotide-frequency**: Analyzes the frequencies of nucleotides and dinucleotides in segments of each label.
- **segtools-overlap**: Analyzes the frequency with which each segment label overlaps features of various types.
- **segtools-html-report**: Combines the output of the other commands and generates an html report for easier viewing.

Utility commands:

- **segtools-compare**: Measure base-wise edit distance between two segmentations.
- **segtools-feature-distance**: Reports the distance from each segment to the nearest feature in each of a list of feature files.
- **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-preprocess**: Preprocess segmentation and annotation files into a binary format that can be quickly re-read in future calls to Segtools commands.

Other commands:

- **segtools-gmtk-parameters**: Analyzes GMTK emission parameters and state transitions.

All the above commands respond to `-h` and `--help`, and this will display the most up-to-date 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.

### Common options

The following options are frequently or always supported by Segtools commands:

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

- **-help, -h**
  
  Display up-to-date usage information and options for the command.
-noplot
Perform computation and output tab files but do not generating 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 full table of commands and corresponding modules:

<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>gmtk-parameters</td>
<td>gmtk_parameters</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>signal-distribution</td>
<td>signal_distribution</td>
</tr>
<tr>
<td>transition</td>
<td>transition</td>
</tr>
</tbody>
</table>

1.5 Commands

1.5.1 segtools-compare

segtools-compare [OPTIONS] SEGMENTATION SEGMENTATION

This command compares two segmentations by a specified metric. Currently, the only supported metric is --edit-distance.
Selected options:

- `--edit-distance, -d`
  Prints (to stout) the base-wise edit distance between two segmentations. This is the number of bases that are assigned different labels in the two segmentations.

### 1.5.2 `segtools-preprocess`

**segtools-preprocess [OPTIONS] INFILE**

This command takes a segmentation or annotation 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.

### 1.5.3 `segtools-aggregation`

**segtools-aggregation [OPTIONS] SEGMENTATION ANNOTATIONS**

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 had two different classes of TSSs that you were interested in (say, expressed and unexpressed), you can use the 3rd 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.

Selected options:

- `--help, -h`
  Display complete usage information

- `--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), 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 than 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, 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. These two models are laid out in logical progressions so that genes are viewed in profile and gene-component-specific associations can be easily seen. Because introns and exons are typically different lengths, --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, 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:

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

where \( f_{\text{obs}} \) is the observed overlap frequency and \( f_{\text{obs}} \) is the overlap frequency expected at random, defined by:

\[
\begin{align*}
f_{\text{obs}} &= \frac{\text{count}(l, p)}{\sum_{\text{labels}} \text{count}(p)} \\
f_{\text{rand}} &= \frac{\text{bases in label}(l)}{\sum_{\text{labels}} \text{bases in label}}
\end{align*}
\]

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 3rd column of the GFF or GTF file (the name field). This is useful if you wanted to compare the aggregation profiles with respect to multiple classes of features, such as TSSs split by expression level or cell type.

-significance
This option includes the significance of the overlap at a region in the plot. If --groups is not specified or there is only one group, then significance is shown by shading the regions that are significant. Otherwise, the significance of the various groups are shown using colored “rugs” at the bottom of the plot. The probability of observing \( n \) overlapping segments of a given label at a given position is modeled with a binomial distribution:

\[
p = \text{binom}(n, N, f_{\text{rand}}),
\]

where \( N \) is the total number of overlapping segments at that position and \( f_{\text{rand}} \) is the same as in --normalize. The p-value is then the probability of observing an overlap count as extreme or more extreme than \( n \) (either enrichment or depletion). This corresponds to a two-tailed binomial test. These pvalues are then transformed to qvalues using Storey et al.’s QVALUE R package, which should be installed if you use this option. At the moment, it doesn’t appear that this significance measure is stringent enough, so use extreme caution when interpreting the results of this option.

Note: If \( n > 100 \) and \( f_{\text{rand}} * N < 10 \), a Poisson approximation is used.

1.5.4 segtools-feature-distance

segtools-feature-distance [OPTIONS] SEGMENTATION ANNOTATIONS...

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

```
chrom<TAB>start<TAB>end<TAB>label<TAB>...
```

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.

1.5.5 segtools-flatten

`segtools-flatten [OPTIONS] SEGMENTATION...`

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 corresponds to the combination of labels for segments that overlap this segment.

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:

```
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 stout, by default, but `--mnemonic-file` and `--outfile` can be specified to create a segmentation file with a corresponding Mnemonics that can be used in further Segtools analyses.

1.5.6 segtools-gmtk-parameters

`segtools-gmtk-parameters [OPTIONS] PARAMSFILE`

This command analyzes the dynamic Bayesian network emission parameters generated by GMTK. This is most useful with segmentations generated using the Segway framework, created by Michael Hoffman. This command just calls the relevant parts of other commands, generating transition plots, a transition graph, and a heatmap of the mean and standard deviation values for each label and track. See `--help` for more information.

1.5.7 segtools-html-report

`segtools-html-report [OPTIONS] SEGMENTATION`

This command is intended to be run after other Segtools commands. Starting in the current working directory (or directory provided with `--results-dir`), it 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.

1.5.8 segtools-transition

`segtools-transition [OPTIONS] SEGMENTATION`

This command takes a segmentation and looks at the transitions between segment labels. In other words, if a segment with label A is directly adjacent to a segment with label B, 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 is just a set of peak calls or regions of interest, 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 connect. This can make it hard to interpret, so the transition frequencies can be thresholded by value (--prob-threshold) or quantile (--quantile-threshold) in drawing the graph.

### 1.5.9 segtools-length-distribution

#### segtools-length-distribution [OPTIONS] SEGMENTATION

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 segmentation, and a table with useful information such as the number of segments of each label, the mean and median segment lengths, and the number of bases covered by each label.

**Note:** This command requires only a segmentation as a parameter and performs minimal computation. As such, it is a useful test to make sure Segtools works on your system.

**Warning:** Since the violin plot is based upon a density distribution, the lengths of all the segments in the segmentation is saved in a tab file to allow this plot to be regenerated solely from R. Unfortunately, for large segmentations, 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.

### 1.5.10 segtools-nucleotide-frequency

#### segtools-nucleotide-frequency [OPTIONS] SEGMENTATION GENOMEDATAFILE

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. CpG content is likely the most interesting output, but nucleotide frequencies can be informative as well.

As input, it takes a segmentation and a Genomedata archive for the genome the segmentation covers. The Genomedata archive is used solely for the nucleotide sequence.

### 1.5.11 segtools-overlap

#### segtools-overlap [OPTIONS] SEGMENTATION ANNOTATIONS

This command measures the base-wise or segment-wise overlap between segments in a segmentation and other annotations. Segments are classified by their label and annotations can be classified with a group, so the basic 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.

**Selected options:**

- **-by <mode>, --by <mode>**
  This specifies whether the overlap analysis will be base-wise (<mode> = "bases") or segment-wise (<mode> = "segments").

- **-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.
**Warning:** The precision-recall plot is accurate for base-wise overlap, but is a rough approximation for segment-wise overlap. Use such results with caution.

### 1.5.12 segtools-signal-distribution

**segtools-signal-distribution [OPTIONS] SEGMENTATION GENOMEDATAFILE**

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.

**Warning:** 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, you merge the results and generate the final output. Here is sample pseudocode:

```python
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>
```

**Selected options:**

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

- **-create-mnemonics**
  This uses 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>
  This loads 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.

### 1.6 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 be two or three tab-separated columns and must contain start with the following header (the description column is optional):
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). Since the new name must fit into small spaces in plots (such as axis labels), it is recommended for this field to be 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 realized you wanted 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, you should construct the following mnemonic file:

```
old  new
A    A1
C    A2
B    B
```

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 command’s `--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 `<segway-users@uw.edu>` mailing list, rather than writing the authors directly. Using the mailing list will get your question answered more quickly. It also allows us to pool knowledge and reduce getting the same inquiries over and over. You can subscribe here:

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

Specifically, 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 Segway, but would like to hear about new releases and other important information, please subscribe to `<segway-announce@uw.edu>` by visiting this web page:

https://mailman1.u.washington.edu/mailman/listinfo/segtools-announce
CHAPTER TWO

INDICES AND TABLES

- genindex
- modindex
- search
INDEX

Symbols
–by <mode>, -b <mode>
segtools-overlap command line option, 10
–chrom <chrom>, -c <chrom>
segtools-signal-distribution command line option, 11
–clobber
command line option, 5
–create-mnemonics
segtools-signal-distribution command line option, 11
–edit-distance, -d
segtools-compare command line option, 7
–groups
segtools-aggregation command line option, 8
–help, -h
command line option, 5
segtools-aggregation command line option, 7
–indir <dir>, -i <dir>
segtools-signal-distribution command line option, 11
–min-overlap <n>
segtools-overlap command line option, 10
–mnemonic-file <file>, -m <file>
command line option, 6
–mode <mode>
segtools-aggregation command line option, 7
–noplot
command line option, 5
–normalize
segtools-aggregation command line option, 8
–outdir <dir>, -o <dir>
command line option, 6
–quick
command line option, 6
–quiet, -q
command line option, 6
–replot
command line option, 6
–significance
segtools-aggregation command line option, 8
–version
command line option, 6

C
command line option
–clobber, 5
–help, -h, 5
–mnemonic-file <file>, -m <file>, 6
–noplot, 5
–outdir <dir>, -o <dir>, 6
–quick, 6
–quiet, -q, 6
–replot, 6
–version, 6

S
segtools-aggregation command line option
–groups, 8
–help, -h, 7
–mode <mode>, 7
–normalize, 8
–significance, 8
segtools-compare command line option
–edit-distance, -d, 7
segtools-overlap command line option
–by <mode>, -b <mode>, 10
–min-overlap <n>, 10
segtools-signal-distribution command line option
–chrom <chrom>, -c <chrom>, 11
–create-mnemonics, 11
–indir <dir>, -i <dir>, 11