Wub package documentation

Release 0.3.1

ONT Applications Group

Sep 03, 2019

Contents

CHAPTER 1

Command line tools

1.1 Command line tools

1.1.1 _template_script

Template script.

```
usage: _template_script [-h] [-i input]
```

Named Arguments

-i Input.

1.1.2 add_errors

Add a specified number of errors to random sites for each input sequence.

```
usage: add_errors [-h] [-n nr_errors] [-t error_type]
                  [input_fasta] [output_fasta]
```

Positional Arguments

input_fasta Input fasta (default: stdin).

Default: <open file '<stdin>', mode 'r' at 0x7f28cd4e80c0>

output_fasta Output fasta (default: stdout)

Default: <open file '<stdout>', mode 'w' at 0x7f28cd4e8150>

Named Arguments

-n Number of errors to introduce (0).

Default: 0

-t Possible choices: substitution, insertion, deletion

Error type: substitution, insertion or deletion.

Default: “substitution”

1.1.3 annotate_length

Add sequence length to sequence record descriptions.

```
usage: annotate_length [-h] [-i in_format] [-o out_format]
                       [input_fastx] [output_fastx]
```

Positional Arguments

input_fastx Input file (default: stdin).

Default: <open file ‘<stdin>’, mode ‘r’ at 0x7f28cd4e80c0>

output_fastx Output file (default: stdout).

Default: <open file ‘<stdout>’, mode ‘w’ at 0x7f28cd4e8150>

Named Arguments

-i Input format (fastq).

Default: “fastq”

-o Output format (fastq).

Default: “fastq”

1.1.4 bam_accuracy

Produce accuracy statistics of the input BAM file. Calculates global accuracy and identity and various per-read statistics.
The input BAM file must be sorted by coordinates and indexed.

```
usage: bam_accuracy [-h] [-c region] [-g global_tsv] [-l read_tsv]
                    [-t bam_tag] [-q aqual] [-e] [-r report_pdf]
                    [-p results_pickle] [-Q]
                    bam
```

Positional Arguments

bam Input BAM file.

Named Arguments

-c	BAM region (None).
-g	Tab separated file to save global statistics (None).
-l	Tab separated file to save per-read statistics (None).
-t	Dataset tag (BAM basename).
-q	Minimum alignment quality (0). Default: 0
-e	Include hard and soft clipps in alignment length when calculating accuracy (False). Default: False
-r	Report PDF (bam_accuracy.pdf). Default: "bam_accuracy.pdf"
-p	Save pickled results in this file (None).
-Q	Be quiet and do not print progress bar (False). Default: False

1.1.5 bam_alignment_length

Produce a tab separated file of alignment lengths and other information. Rows are sorted by number of aligned reference bases unless the -x option is specified.

```
usage: bam_alignment_length [-h] [-t tsv_file] [-q aqual] [-x] [-Q] bam
```

Positional Arguments

bam	Input BAM file.

Named Arguments

-t	Tab separated file to save alignment lengths (bam_alignment_length.tsv). Default: "bam_alignment_length.tsv"
-q	Minimum alignment quality (0). Default: 0
-x	Sort by number of read bases instead of number of aligned reference bases. Default: False
-Q	Be quiet and do not print progress bar (False). Default: False

1.1.6 bam_alignment_qc

Produce alignment based QC plots of the input BAM file. The input BAM file must be sorted by coordinates and indexed.

It produces the following global plots:

- Read statistics: number of mapped, unmapped and low mapping quality reads.
- Distribution of mean quality values in the mapped and unmapped fractions.
- Distribution of read lengths in the unmapped fraction.
- Distribution of read lengths in the mapped fraction.
- Distribution of read lengths in the mapping with quality less than -q
- Distribution of alignment lengths.
- Distribution of mapping qualities.
- Plot of alignment lengths vs. mean base qualities.
- Basewise statistics: total alignment length, number of insertions, deleltions, matches and mismatches.
- Precision statistics: accuracy and identity.
- Frequency of errors in the context specifed by the left and right context sizes (-n). Definition of context: for substitutions the event is happening from the "central base", in the case of indels the events are located between the central base and the base before. The columns of the heatmap are normalised to sum to one and then the diagonal element are set to zero.
- Distribution of deletion lengths.
- Distribution of insertion lengths.
- Base composition of insertions.

The following plots are produced for every reference unless disabled via -x:

- Distribution of quality values across the reference as a heatmap.
- Mean quality values across the reference.
- Base coverage across the reference.

The tool saves the gathered statistics in a pickle file, which can be fed to *bam_multi_qc.py* to compare different samples.

```
usage: bam_alignment_qc [-h] -f reference [-c region] [-n context_sizes] [-x]
                        [-t bam_tag] [-q aqual] [-i qual_ints] [-r report_pdf]
                        [-p results_pickle] [-Q]
                        bam
```

Positional Arguments

bam	Input BAM file.

Named Arguments

-f	Reference fasta.
-c	BAM region (None).

-n	Left and right context sizes (1,1). Default: "1,1"
-x	Do not plot per-reference information. Default: False
-t	Dataset tag (BAM basename).
-q	Minimum alignment quality (0). Default: 0
-i	Number of quality intervals (6). Default: 6
-r	Report PDF (bam_alignment_qc.pdf). Default: "bam_alignment_qc.pdf"
-p	Save pickled results in this file (bam_alignment_qc.pk). Default: "bam_alignment_qc.pk"
-Q	Be quiet and do not show progress bars. Default: False

1.1.7 bam_alignments_compare

Compare alignments stored in two BAM files. The two BAM files must have the same set of reads in the same order (name sorted).

```
usage: bam_alignments_compare [-h] [-w coarse_tolerance] [-g] [-r report_pdf]
                              [-p results_pickle] [-t tsv_file] [-f format]
                              [-Q]
                              bam_one bam_two
```

Positional Arguments

bam_one	First input BAM file.
bam_two	Second input BAM file.

Named Arguments

-w	Tolerance when performing coarse comparison of alignments (50). Default: 50
-g	Do strict comparison of alignment flags. Default: False
-r	Report PDF (bam_alignments_compare.pdf). Default: "bam_alignments_compare.pdf"
-p	Save pickled results in this file (bam_alignments_compare.pk). Default: "bam_alignments_compare.pk"

-t	Save results in tsv format in this file (None).
-f	Input format (BAM).
	Default: "BAM"
-Q	Be quiet and do not print progress bar (False).
	Default: False

1.1.8 bam_count_reads

Count reads mapping to each reference in a BAM file.

```
usage: bam_count_reads [-h] [-a min_aqual] [-f in_format] [-z ref_fasta]
                       [-k words] [-g] [-p results_pickle] [-t tsv_file] [-Q]
                       [-R] [-F yield_freq]
                       [bam]
```

Positional Arguments

bam	Input file (default: stdin).
	Default: <open file '<stdin>', mode 'r' at 0x7f28cd4e80c0>

Named Arguments

-a	Minimum mapping quality (0).
	Default: 0
-f	Input format (BAM).
	Default: "BAM"
-z	Reference fasta. GC content and length columns are added if present (None).
-k	Include word frequencies of specifed length in output (None).
-g	Include mean GC content of reads mapped to each reference (False).
	Default: False
-p	Save pickled results in this file (None).
-t	Save results in tsv format in this file (bam_count_reads.tsv).
	Default: "bam_count_reads.tsv"
-Q	Be quiet and do not print progress bar (False).
	Default: False
-R	Count reads from SAM stream in stdin. Only read count fields are written. Header required! (False).
	Default: False
-F	Yield counts after every -Fth mapped record when doing online counting (100).
	Default: 100

1.1.9 bam_cov

Produce refrence coverage table.

```
usage: bam_cov [-h] -f reference [-c region] [-t tsv] [-q aqual] [-Q] bam
```

Positional Arguments

bam	Input BAM file.

Named Arguments

-f	Reference fasta.
-c	BAM region (None).
-t	Output TSV (bam_cov.tsv).
	Default: "bam_cov.tsv"
-q	Minimum alignment quality (0).
	Default: 0
-Q	Be quiet and do not show progress bars.
	Default: False

1.1.10 bam_fill_unaligned

Generate SAM records for the reads present in the input fastq but missing from the input SAM/BAM.

```
usage: bam_fill_unaligned [-h] [-f format] -q fastq input_file output_file
```

Positional Arguments

input_file	Input file.
output_file	Output SAM file.

Named Arguments

-f	Input/output format (SAM).
	Default: "SAM"
-q	Input fastq.

1.1.11 bam_frag_coverage

Produce aggregated and individual plots of fragment coverage.

```
usage: bam_frag_coverage [-h] -f reference [-c region] [-i intervals]
                         [-b bins] [-x] [-o] [-t bam_tag] [-q aqual]
                         [-l cov80_tsv] [-g glob_cov80_tsv] [-r report_pdf]
                         [-p results_pickle] [-Q]
                         bam
```

Positional Arguments

bam Input BAM file.

Named Arguments

-f Reference fasta.

-c BAM region (None).

-i Length intervals ().

Default: “”

-b Number of bins (None = auto).

-x Plot per-reference information.

Default: False

-o Do not take log of coverage.

Default: False

-t Dataset tag (BAM basename).

-q Minimum alignment quality (0).

Default: 0

-l Tab separated file with per-chromosome cov80 scores (None). Requires the -x option to be specified.

-g Tab separated file with global cov80 score (None).

-r Report PDF (bam_frag_coverage.pdf).

Default: “bam_frag_coverage.pdf”

-p Save pickled results in this file (None).

-Q Be quiet and do not show progress bars.

Default: False

1.1.12 bam_gc_vs_qual

Produce a plot of GC content of aligned read and reference portion versus their mean quality values.

```
usage: bam_gc_vs_qual [-h] -f reference [-q aqual] [-r report_pdf] [-t tsv]
                      [-Q]
                      bam
```

Positional Arguments

bam	Input BAM file.

Named Arguments

-f	Reference fasta.
-q	Minimum alignment quality (0).
	Default: 0
-r	Report PDF (bam_gc_vs_qual.pdf).
	Default: "bam_gc_vs_qual.pdf"
-t	Tab separated file to save results (bam_gc_vs_qual.tsv).
	Default: "bam_gc_vs_qual.tsv"
-Q	Be quiet and do not show progress bars.
	Default: False

1.1.13 bam_multi_qc

Compare alignment QC statistics of multiple samples.

It takes a list of pickle files produced by *bam_alignment_qc.py* and produces plots comparing the following properties of the input samples:

- Number of mapped reads.
- Number of unmapped reads.
- Distribution of mean quality values in the unaligned fraction.
- Distribution of mean quality values in the aligned fraction.
- Distribution of read lengths in the unaligned fraction.
- Distribution of read lengths in the aligned fraction.
- Distribution of alignment lengths.
- Distribution of mapping qualities.
- Alignment accuracy.
- Alignment identity.
- Distribution of deletion lengths.
- Distribution of insertion lengths.

Per reference plots (can be disabled by -x):

- Relative coverage across reference.
- Mean qualities per position.

```
usage: bam_multi_qc [-h] [-r report_pdf] [-x]
                    [input_pickles [input_pickles ...]]
```

Positional Arguments

input_pickles Input pickles.

Named Arguments

-r Report PDF (bam_multi_qc.pdf).
Default: "bam_multi_qc.pdf"

-x Do not plot reference statistics.
Default: False

1.1.14 bam_ref_base_coverage

Calculate percent covered reference lengths.

```
usage: bam_ref_base_coverage [-h] -f reference [-c region] [-t tsv]
                             [-m min_cov] [-Q]
                             bam
```

Positional Arguments

bam Input BAM file.

Named Arguments

-f Reference fasta.

-c BAM region (None).

-t Output tab separated file (bam_ref_base_coverage.tsv).
Default: "bam_ref_base_coverage.tsv"

-m Minimum base coverage for a position to be counted (1).
Default: 1

-Q Be quiet and do not show progress bars.
Default: False

1.1.15 bam_ref_tab

Produce a tab separated file with read identifiers and the corresponding references, sorted by reference.

```
usage: bam_ref_tab [-h] [-t read_tsv] [-Q] [-s] bam
```

Positional Arguments

bam Input BAM file.

Named Arguments

-t Tab separated file to save reference table.

Default: "bam_ref_tab.tsv"

-Q Be quiet and do not print progress bar (False).

Default: False

-s Save read strand in output (False).

Default: False

1.1.16 bam_score_filter

Filter SAM/BAM records by score or other criteria. WARNING: the input records must be sorted by name or the filtering will not work as expected.

```
usage: bam_score_filter [-h] [-f format] [-s strategy] [-q query_cover]
                        input_file output_file
```

Positional Arguments

input_file Input file.

output_file Output SAM file.

Named Arguments

-f Input/output format (SAM).

Default: "SAM"

-s Possible choices: top_per_query, query_coverage, ref_coverage

Filtering strategy: top_per_query, query_coverage, ref_coverage (top_per_query).

Default: "top_per_query"

-q Minimum query coverage fraction (0.8).

Default: 0.8

1.1.17 bam_soft_clips_tab

Produce a tab separated file with read identifiers and number of soft clipped bases at each end (relative to the original sequence in the fastq).

```
usage: bam_soft_clips_tab [-h] [-t tsv] [-Q] bam
```

Positional Arguments

bam Input BAM file.

Named Arguments

-t Output tab separated file.

Default: "bam_soft_clips_tab.tsv"

-Q Be quiet and do not print progress bar (False).

Default: False

1.1.18 bias_explorer

Simple tool for exploring biases in transcript counts. Takes as input count files generated by bam_count_reads.py (with the -z flag) and performs linear regression of log counts against transcript length and GC content.

```
usage: bias_explorer [-h] [-r report_pdf] [-x] count_file
```

Positional Arguments

count_file Input counts file with length ang GC content features.

Named Arguments

-r Report PDF (bias_explorer.pdf).

Default: "bias_explorer.pdf"

-x Exclude transcripts with zero counts.

Default: False

1.1.19 calculate_coverage

Calculate total number of bases and genome coverage if genome size is given.

```
usage: calculate_coverage [-h] [-f format] [-s genome_size]
                          [-p results_pickle]
                          [input_fastx]
```

Positional Arguments

input_fastx Input (default: stdin).

Default: <open file '<stdin>', mode 'r' at 0x7f28cd4e80c0>

Named Arguments

-f Input format (fastq).

Default: "fastq"

-s Genome size (None).

-p Save pickled results in this file.

1.1.20 compare_genomes_dnadiff

Compare a set of reference sequences (genome) to another set (target assembly) using mummer's dnadiff. It prints the alignment results to stdout. All parsed results can be saved in a pickle file.

```
usage: compare_genomes_dnadiff [-h] [-p results_pickle] [-r raw_file]
                               [-d work_dir] [-k] [-v]
                               reference_fasta target_fasta
```

Positional Arguments

reference_fasta Reference fasta.

target_fasta Target fasta.

Named Arguments

-p Save pickled results in this file (None).

-r Save dnadiff report in this file (None).

-d Use this working directory instead of a temporary directory (None).

-k Keep dnadiff result files (False).

Default: False

-v Print out dnadiff output (False).

Default: False

1.1.21 compare_genomes_lastal

Compare a set of reference sequences (genome) to another set (target assembly) using lastal alignment. Accuracy is the total number of matched bases divided by total alignment length. Coverage is total reference covered by alignment divided by total length of reference.

Caveats:

- The lastal alignments are filtered by default (use -f to disable) so only the best scoring alignment is kept per query. Hence some shorter valid

alignments might be discarded causing an underestimation of coverage. - The estimated accuracy is dependent on the scoring of gaps and mismatches. By default gap open and gap extend penalties are set to equal.

```
usage: compare_genomes_lastal [-h] [-p results_pickle] [-l lastal_args]
                              [-t details_tsv] [-f] [-r report_pdf]
                              reference_fasta target_fasta
```

Positional Arguments

reference_fasta Reference fasta.

target_fasta Target fasta.

Named Arguments

-p Save pickled results in this file (None).

-l Parameters passed to lastal in the <arg>:value,... format (a:1,b:1).

Default: "a:1,b:1"

-t Save details of lastal alignment in this tab-separated file (None).

-f Do *not* filter for best alignment per query.

Default: False

-r Report with alignment details plot (None).

1.1.22 convert_alphabet

Convert between DNA and RNA alphabets.

```
usage: convert_alphabet [-h] [-i in_format] [-o out_format] [-D] [-R]
                        [input_fastx] [output_fastx]
```

Positional Arguments

input_fastx Input file (default: stdin).

Default: <open file '<stdin>', mode 'r' at 0x7f28cd4e80c0>

output_fastx Output file (default: stdout).

Default: <open file '<stdout>', mode 'w' at 0x7f28cd4e8150>

Named Arguments

-i Input format (fastq).

Default: "fastq"

-o Output format (fastq).

Default: "fastq"

-D RNA->DNA alphabet conversion.

Default: False

-R DNA->RNA alphabet conversion.

Default: False

1.1.23 correlate_counts

Correlate counts produced by multiple runs of bam_count_reads.py.

```
usage: correlate_counts [-h] [-r report_pdf] [-c corr_type] [-L] [-o]
                        [input_counts [input_counts ...]]
```

Positional Arguments

input_counts Input counts as tab separated files.

Named Arguments

-r Report PDF (bam_multi_qc.pdf).

Default: "correlate_counts.pdf"

-c Correlation statistic - spearman or pearson (spearman).

Default: "spearman"

-L Log transform data.

Default: False

-o Omit lower diagonal.

Default: False

1.1.24 fasta_to_mock_fastq

Convert fasta file to fastq with mock qualities.

```
usage: fasta_to_mock_fastq [-h] [-q mock_quals] [input_fasta] [output_fastq]
```

Positional Arguments

input_fasta Input fasta (default: stdin).

Default: <open file '<stdin>', mode 'r' at 0x7f28cd4e80c0>

output_fastq Output fastq (default: stdout)

Default: <open file '<stdout>', mode 'w' at 0x7f28cd4e8150>

Named Arguments

-q Mock quality value (40).

Default: 40

1.1.25 fastq_qual_tab

Generate a table of read names and mean quality values.

```
usage: fastq_qual_tab [-h] [-t tsv] [input_fastq]
```

Positional Arguments

input_fastq Input fastq (default: stdin).

Default: <open file '<stdin>', mode 'r' at 0x7f28cd4e80c0>

Named Arguments

-t Output tab separated file.

Default: "fastq_qual_tab.tsv"

1.1.26 fastq_time_slice

Filter a fastq file by starting time.

```
usage: fastq_time_slice [-h] -t time_tsv [-s start_perc] [-e end_perc]
                        [input_fastq] [output_fastq]
```

Positional Arguments

input_fastq Input fastq (default: stdin).

Default: <open file '<stdin>', mode 'r' at 0x7f28cd4e80c0>

output_fastq Output fastq (default: stdout)

Default: <open file '<stdout>', mode 'w' at 0x7f28cd4e8150>

Named Arguments

-t Tab separeted file produced by fastq_time_tab.py.

-s Start of slice as percent of total time.

Default: 0.0

-e End of slice as percent of total time.

Default: 100.0

1.1.27 fastq_time_tab

Produce a tab separated file with read start times, read and channel numbers sorted by start time.

```
usage: fastq_time_tab [-h] [-t read_tsv] fastq
```

Positional Arguments

fastq Input fastq file.

Named Arguments

-t Tab separated file to save read time table.

Default: "fastq_time_tab.tsv"

1.1.28 fastx_ends_tab

Generate a tab separated file with the first and last -n bases of the sequences.

```
usage: fastx_ends_tab [-h] [-i in_format] [-n nr_bases]
                      [input_fastx] [output_tsv]
```

Positional Arguments

input_fastx Input file (default: stdin).

Default: <open file '<stdin>', mode 'r' at 0x7f28cd4e80c0>

output_tsv Output file (default: stdout).

Default: <open file '<stdout>', mode 'w' at 0x7f28cd4e8150>

Named Arguments

-i Input format (fastq).

Default: "fastq"

-n .

Default: 100

1.1.29 fastx_grep

Filter sequence files by read name.

```
usage: fastx_grep [-h] [-i in_format] [-o out_format] [-n read_names]
                  [input_fastx] [output_fastx]
```

Positional Arguments

input_fastx Input file (default: stdin).

Default: <open file '<stdin>', mode 'r' at 0x7f28cd4e80c0>

output_fastx Output file (default: stdout).

Default: <open file '<stdout>', mode 'w' at 0x7f28cd4e8150>

Named Arguments

-i Input format (fastq).
Default: "fastq"

-o Output format (fastq).
Default: "fastq"

-n Comma separated list of read names to select.
Default: ""

1.1.30 fastx_length_tab

Generate a tab separated file with the sequence lengths in the input file.

```
usage: fastx_length_tab [-h] [-i in_format] [input_fastx] [output_tsv]
```

Positional Arguments

input_fastx Input file (default: stdin).
Default: <open file '<stdin>', mode 'r' at 0x7f28cd4e80c0>

output_tsv Output file (default: stdout).
Default: <open file '<stdout>', mode 'w' at 0x7f28cd4e8150>

Named Arguments

-i Input format (fasta).
Default: "fasta"

1.1.31 length_normalise_counts

Calculate RPKM values from raw counts and a transcriptome reference.

```
usage: length_normalise_counts [-h] -f in_trs input_counts output_count
```

Positional Arguments

input_counts Input count file.

output_count Output RPKM file.

Named Arguments

-f Input transcriptome.

1.1.32 merge_tsvs

Merge tab separated files on a given field using pandas.

```
usage: merge_tsvs [-h] [-j join] [-f field] [-o out_tsv] [-z]
                  [input_tsvs [input_tsvs ...]]
```

Positional Arguments

input_tsvs Input tab separated files.

Named Arguments

-j Join type (outer).

Default: "outer"

-f Join on this field (Read).

Default: "Read"

-o Output tsv (merge_tsvs.tsv).

Default: "merge_tsvs.tsv"

-z Fill NA values with zero.

Default: False

1.1.33 multi_length_hist

Plot histograms of length distributions from multiple sequence files.

```
usage: multi_length_hist [-h] [-r report_pdf] [-f in_format] [-b nr_bins]
                         [-l min_len] [-u max_len] [-L]
                         [input_counts [input_counts ...]]
```

Positional Arguments

input_counts Input sequence files.

Named Arguments

-r Report PDF.

Default: "multi_length_hist.pdf"

-f Input format (fastq).

Default: "fastq"

-b Number of bins (50).

Default: 50

-l Minimum read length (None).

-u	Maximum read length (None).
-L	Log transform lengths. Default: False

1.1.34 pickle_cat

Pretty print the contents of a pickle file.

```
usage: pickle_cat [-h] pickle_file
```

Positional Arguments

pickle_file	Input pickle file.

1.1.35 plot_counts_correlation

Scatter plot of two set of counts.

```
usage: plot_counts_correlation [-h] [-r report_pdf] [-T tags] [-t merged_data]
                               [-o Correlation_tsv]
                               counts_one counts_two
```

Positional Arguments

counts_one	Input tab separated file.
counts_two	Input tab separated file.

Named Arguments

-r	Report PDF. Default: "plot_counts_correlation.pdf"
-T	Data tags: tag1,tag2.
-t	Merged data TSV.
-o	Correlation TSV.

1.1.36 plot_gffcmp_stats

Plot a gffcompare stats file.

```
usage: plot_gffcmp_stats [-h] [-r report_pdf] [-p pickle_out] input_txt
```

Positional Arguments

input_txt	Input gffcompare stats file.

Named Arguments

-r Report PDF (plot_gffcmp_stats.pdf).

Default: "plot_gffcmp_stats.pdf"

-p Output pickle file.

Default: "plot_gffcmp_stats.pk"

1.1.37 plot_qualities

Plot the mean quality values across non-overlapping windows in the input sequences.

```
usage: plot_qualities [-h] [-w win_size] [-r report_pdf] [input_fastx]
```

Positional Arguments

input_fastx Input (default: stdin).

Default: <open file '<stdin>', mode 'r' at 0x7f28cd4e80c0>

Named Arguments

-w Window size (50).

Default: 50

-r Report pdf (plot_qualities.pdf).

Default: "plot_qualities.pdf"

1.1.38 plot_sequence_properties

Plot histograms of lengths and quality values.

```
usage: plot_sequence_properties [-h] [-f format] [-b bins] [-r report_pdf]
                                [-j]
                                [input_fastx]
```

Positional Arguments

input_fastx Input (default: stdin).

Default: <open file '<stdin>', mode 'r' at 0x7f28cd4e80c0>

Named Arguments

-f Input format (fastq).

Default: "fastq"

-b Number of bins on histograms (50).

Default: 50

-r	Report pdf (plot_sequence_properties.pdf).
	Default: "plot_sequence_properties.pdf"
-j	Produce joint plot of lengths and mean quality values (False).
	Default: False

1.1.39 reads_across_time

Plot read and alignment properties across time.

```
usage: reads_across_time [-h] -i time_tab -a aln_tab [-w res_freq]
                         [-r report_pdf] [-t out_tsv]
```

Named Arguments

-i	Tab separated file generated by fastq_time_tab.py
-a	Tab separated file generated by bam_alignment_length.py
-w	Resampling frequency in minutes.
	Default: 5
-r	Report PDF (reads_across_time.pdf).
	Default: "reads_across_time.pdf"
-t	Output tsv (reads_across_time.tsv).
	Default: "reads_across_time.tsv"

1.1.40 reads_stats

No documentation available .. _reverse_fastq:

1.1.41 reverse_fastq

Reverse (but not complement!) sequences and qualities in fastq file.

```
usage: reverse_fastq [-h] [input_fastq] [output_fastq]
```

Positional Arguments

input_fastq	Input fastq (default: stdin).
	Default: <open file '<stdin>', mode 'r' at 0x7f28cd4e80c0>
output_fastq	Output fastq (default: stdout)
	Default: <open file '<stdout>', mode 'w' at 0x7f28cd4e8150>

1.1.42 sequence_filter

Filter sequences by length and mean quality value.

```
usage: sequence_filter [-h] [-i in_format] [-o out_format] [-q min_qual]
                       [-l min_length] [-c] [-u max_length]
                       [input_fastx] [output_fastx]
```

Positional Arguments

input_fastx Input file (default: stdin).

Default: <open file '<stdin>', mode 'r' at 0x7f28cd4e80c0>

output_fastx Output file (default: stdout).

Default: <open file '<stdout>', mode 'w' at 0x7f28cd4e8150>

Named Arguments

-i Input format (fastq).

Default: "fastq"

-o Output format (fastq).

Default: "fastq"

-q Minimum mean quality value (0.0).

Default: 0.0

-l Minimum length (0).

Default: 0

-c Reverse complement sequences.

Default: False

-u Maximum length (None).

1.1.43 sequence_subtract

Filter out sequences present in the first file from the second file.

```
usage: sequence_subtract [-h] [-i in_format] [-o out_format]
                         [input_fastx_bait] [input_fastx_target]
                         [output_fastx]
```

Positional Arguments

input_fastx_bait First input file (default: stdin).

Default: <open file '<stdin>', mode 'r' at 0x7f28cd4e80c0>

input_fastx_target Second input file.

Default: <open file '<stdin>', mode 'r' at 0x7f28cd4e80c0>

output_fastx Output file (default: stdout).

Default: <open file '<stdout>', mode 'w' at 0x7f28cd4e8150>

Named Arguments

-i Input format (fastq).

Default: "fastq"

-o Output format (fastq).

Default: "fastq"

1.1.44 simulate_errors

Simulate sequencing errors for each input sequence.

```
usage: simulate_errors [-h] [-e error_rate] [-w error_weights]
                       [-z random_seed]
                       [input_fasta] [output_fasta]
```

Positional Arguments

input_fasta Input fasta (default: stdin).

Default: <open file '<stdin>', mode 'r' at 0x7f28cd4e80c0>

output_fasta Output fasta (default: stdout)

Default: <open file '<stdout>', mode 'w' at 0x7f28cd4e8150>

Named Arguments

-e Total rate of substitutions insertions and deletions (0.1).

Default: 0.1

-w Relative frequency of substitutions,insertions,deletions (1,1,4).

Default: "1,1,4"

-z Random seed (None).

1.1.45 simulate_genome

Simulate genome sequence with the specified number of chromosomes, length distribution (truncated gamma) and base composition.

```
usage: simulate_genome [-h] [-n nr_chrom] [-m mean_length] [-a gamma_shape]
                       [-l low_trunc] [-u high_trunc] [-b base_freqs]
                       [-z random_seed]
                       [output_fasta]
```

Positional Arguments

output_fasta Output fasta (default: stdout)

Default: <open file '<stdout>', mode 'w' at 0x7f28cd4e8150>

Named Arguments

-n Number of chromosomes (23).

Default: 23

-m Mean length of chromosomes (5000000).

Default: 5000000

-a Gamma shape parameter (1).

Default: 1.0

-l Lower truncation point (None).

-u Upper truncation point (None).

-b Relative base frequencies in A,C,G,T order (1,1,1,1) or "random".

Default: "1,1,1,1"

-z Random seed (None).

1.1.46 simulate_sequences

Simulate sequences of fixed length and specified base composition.

```
usage: simulate_sequences [-h] [-n nr_seq] [-m length] [-b base_freqs]
                          [-z random_seed]
                          [output_fasta]
```

Positional Arguments

output_fasta Output fasta (default: stdout)

Default: <open file '<stdout>', mode 'w' at 0x7f28cd4e8150>

Named Arguments

-n Number of sequences (1).

Default: 1

-m Length of simulated sequences (3000).

Default: 3000

-b Relative base frequencies in A,C,G,T order (1,1,1,1).

Default: "1,1,1,1"

-z Random seed (None).

1.1.47 simulate_sequencing_simple

Sample fragments from the input genome and simulate sequencing errors. Read lengths are drawn from the specified truncated gamma distribution. Chromosomes are sampled randomly for each read.

The format of the read names is the following: r<unique_id>_<chromosome>_<frag_start>_<frag_end>_<strand>/q<realised_qua

```
usage: simulate_sequencing_simple [-h] [-n nr_reads] [-m mean_length]
                                  [-a gamma_shape] [-l low_trunc]
                                  [-u high_trunc] [-e error_rate]
                                  [-w error_weights] [-b strand_bias]
                                  [-q mock_quality] [-s true_sam] [-Q]
                                  [-z random_seed]
                                  [input_fasta] [output_fastq]
```

Positional Arguments

input_fasta Input genome in fasta format (default: stdin).

Default: <open file '<stdin>', mode 'r' at 0x7f28cd4e80c0>

output_fastq Output fastq (default: stdout)

Default: <open file '<stdout>', mode 'w' at 0x7f28cd4e8150>

Named Arguments

-n Number of simulated reads (1).

Default: 1

-m Mean read length (5000).

Default: 5000

-a Read length distribution: gamma shape parameter (1).

Default: 1.0

-l Read length distribution: lower truncation point (100).

Default: 100

-u Read length distribution: upper truncation point (None).

-e Total rate of substitutions insertions and deletions (0.1).

Default: 0.1

-w Relative frequency of substitutions,insertions,deletions (1,1,4).

Default: "1,1,4"

-b Strand bias: the ratio of forward and reverse reads (0.5).

Default: 0.5

-q Mock base quality for fastq output (40).

Default: 40

-s Save true alignments in this SAM file (None).

-Q Be quiet and do not print progress bar (False).

Default: False

-z Random seed (None).

1.1.48 split_fastx

Split sequence records in file to one record per file or batches of records.

```
usage: split_fastx [-h] [-i in_format] [-o out_format] [-b batch_size]
                   [input_fastx] [output_dir]
```

Positional Arguments

input_fastx Input file (default: stdin).

Default: <open file '<stdin>', mode 'r' at 0x7f28cd4e80c0>

output_dir Output directory (default: .)

Default: "."

Named Arguments

-i Input format (fastq).

Default: "fastq"

-o Output format (fastq).

Default: "fastq"

-b Batch size (None).

CHAPTER 2

Full API reference

2.1 Usage

To use wub package in a project:

```
import wub
```

2.2 wub

2.2.1 wub package

Subpackages

wub.bam package

Submodules

wub.bam.common module

wub.bam.common.**pysam_open**(*alignment_file*, *in_format='BAM'*)

Open SAM/BAM file using pysam.

Parameters

- **alignment_file** – Input file.
- **in_format** – Format (SAM or BAM).

Returns pysam.AlignmentFile

Return type pysam.AlignmentFile

wub.bam.compare module

Compares alignments in two BAM files.

wub.bam.compare.**aligned_pairs_to_matches**(*aligned_pairs*, *offset*)
Convert aligned pairs into a sequence of reference positions.

Parameters

- **aligned_pairs** – Iterator of aligned pairs.
- **offset** – Offset at the beggining of the sequences.

Returns Iterator of reference positions aligned to the sequences positions.

Return type generator

wub.bam.compare.**bam_compare**(*aln_one*, *aln_two*, *coarse_tolerance=50*, *strict_flags=False*, *in_format='BAM'*, *verbose=False*)
Count reads mapping to references in a BAM file.

Parameters

- **alignment_file** – BAM file.
- **min_aln_qual** – Minimum mapping quality.
- **verbose** – Show progress bar.

Returns Dictionary with read counts per reference.

Return type dict

wub.bam.compare.**calc_consistency_score**(*segment_one*, *segment_two*, *offset_one*, *offset_two*)
Calculate the number of bases aligned to the same reference bases in two alignments. :param segment_one: Pysam aligned segments. :param segment_two: Pysam aligned segments. :param offset_one: Hard clipping offset for the first alignment. :param offset_two: Hard clipping offset for the second alignment. :retruns: Number of matching base alignments. :rtype: int

wub.bam.compare.**compare_alignments**(*segment_one*, *segment_two*, *strict_flags=False*)
Count reads mapping to references in a BAM file.

Parameters

- **alignment_file** – BAM file.
- **min_aln_qual** – Minimum mapping quality.

Returns Dictionary with read counts per reference.

Return type dict

wub.bam.compare.**count_clipped**(*aln*, *target_op*)
Count hard clipped bases in aligned segment.

Parameters

- **aln** – Pysam aligned segement.
- **target_op** – CIGAR operation.

Returns Number of hard clipped bases in segment.

Return type int

wub.bam.compare.**get_hard_clip_offset**(*aln*)
Get hard clipping offset from alignment.

Parameters **aln** – Pysam aligned segment.

Returns Hard clipping offset.

Return type int

wub.bam.compare.**is_coarse_match**(*aln_diff*, *tolerance*)
Determine if start and end postions of two alignments are within the specified tolerance levels.

Parameters **aln_diff** – Alignment diff structure as returned by compare_alignments.

Returns True or False

Return type bool

wub.bam.filter module

Filter SAM/BAM records by various criteria.

wub.bam.filter.**filter_query_coverage**(*records_iter*, *minimum_coverage*)
Filter pysam records keeping the ones with sufficient query coverage.

Parameters

- **records_iter** – Iterator of pysam aligned segments.
- **minimum_coverage** – Minimum fraction of covered query.

Returns Generator of filtered records.

Return type generator

wub.bam.filter.**filter_ref_coverage**(*records_iter*, *minimum_coverage*, *header*)
Filter pysam records keeping the ones with sufficient reference coverage.

Parameters

- **records_iter** – Iterator of pysam aligned segments.
- **minimum_coverage** – Minimum fraction of covered reference.
- **header** – SAM header with reference lengths.

Returns Generator of filtered records.

Return type generator

wub.bam.filter.**filter_top_per_query**(*records_iter*)
Filter pysam records keeping top scoring per query. Assumes records are sorted by name.

Parameters **records_iter** – Iterator of pysam aligned segments.

Returns Generator of filtered records.

Return type generator

wub.bam.filter.**get_alignment_score**(*segement*)
Get alignment score from pysam segment.

Parameters **segment** – Pysam aligned segment.

Returns Alignment score.

Return type int

wub.bam.read_counter module

Count reads per reference in BAM/SAM file.

wub.bam.read_counter.**count_reads**(*alignment_file*, *in_format='BAM'*, *min_aln_qual=0*, *verbose=False*, *reads_gc=False*)

Count reads mapping to references in a BAM file.

Parameters

- **alignment_file** – BAM file.
- **min_aln_qual** – Minimum mapping quality.
- **verbose** – Verbose if True.
- **read_gc** – Calculate mean GC content of reads for each reference.

Returns Dictionary with read counts per reference and read GC contents.

Return type tuple of dicts

wub.bam.read_counter.**count_reads_realtime**(*alignment_file='-'*, *in_format='SAM'*, *min_aln_qual=0*, *yield_freq=1*, *verbose=False*)

Online counting of reads mapping to references in a SAM/BAM stream from stdin.

Parameters

- **alignment_file** – BAM file (stdin).
- **min_aln_qual** – Minimum mapping quality.
- **yield_freq** – Yield frequency.
- **verbose** – Minimum mapping quality.

Returns Generator of dictionary with read counts per reference.

Return type generator

wub.bam.sam_writer module

class wub.bam.sam_writer.**SamWriter**(*out_file*, *header=None*)

Simple class to write SAM files.

Initialise SAM writer object

close()

Close SAM file.

Parameters **self** – object

Returns None

Return type object

new_sam_record(*qname*, *flag*, *rname*, *pos*, *mapq*, *cigar*, *rnext*, *pnext*, *tlen*, *seq*, *qual*, *tags*)

Create new SAM record structure.

Parameters

- **self** – object
- **qname** – Read name.

- **rname** – Reference name.
- **pos** – Position in reference.
- **mapq** – Mapping quality.
- **cigar** – CIGAR string.
- **rnext** – Reference of next read.
- **pnext** – Position of next read.
- **tlen** – Template length.
- **seq** – Read sequence.
- **qual** – Base qualities.
- **tags** – Optional tags.

Returns SAM record.

Return type OrderedDict

write(*record*)

Write SAM record to file.

Parameters

- **self** – object
- **record** – SAM record.

Returns None

Return type object

wub.bam.stats module

wub.bam.stats.**error_and_read_stats**(*bam*, *refs*, *context_sizes=(1, 1)*, *region=None*, *min_aqual=0*, *verbose=True*)

Gather read statistics and context-dependend error statistics from BAM file. WARNING: context overstepping reference start/end boundaries are not registered.

Definition of context: for substitutions the event is happening from the "central base", in the case of indels the events are located between the central base and the base before.

Parameters

- **bam** – Input BAM file.
- **refs** – Dictionary of references.
- **context_sizes** – The size of the left and right contexts.
- **region** – samtools regions.
- **min_qual** – Minimum mappign quality.
- **verbose** – Show progress bar.

Returns Dictionary with read and error statistics.

Return type dict

wub.bam.stats.**frag_coverage**(*bam*, *chrom_lengths*, *region=None*, *min_aqual=0*, *ref_cov=True*, *verbose=True*)

Calculate fragment coverage vectors on the forward and reverse strands.

Parameters

- **bam** – Input bam file.
- **chrom_lengths** – Dictionary of chromosome names and lengths.
- **region** – Restrict parsing to the specified region.
- **min_aqual** – Minimum mapping quality.
- **verbose** – Display progress bar.

Returns Forward and reverse fragment coverage vectors.

Return type dict

wub.bam.stats.**pileup_stats**(*bam*, *region=None*, *verbose=True*, *with_quals=True*)

Parse pileup columns and extract quality values.

Parameters

- **bam** – Input BAM file.
- **region** – samtools region.
- **verbose** – Show progress bar.
- **with_quals** – Return quality values per position.

Returns Dictionaries per reference with per-base coverage and quality values.

Return type dict

wub.bam.stats.**read_stats**(*bam*, *min_aqual=0*, *region=None*, *with_clipps=False*, *verbose=True*)

Parse reads in BAM file and record various statistics.

Parameters

- **bam** – BAM file.
- **min_aqual** – Minimum mapping quality, skip read if mapping quality is lower.
- **region** – smatools region.
- **with_clipps** – Take into account clipps when calculating accuracy.
- **verbose** – Show progress bar.

Returns A dictionary with various global and per-read statistics.

Return type dict

wub.bam.stats.**stats_from_aligned_read**(*read*, *with_clipps=False*)

Create summary information for an aligned read (modified from tang.util.bio).

Parameters

- **read** – `pysam.AlignedSegment` object
- **with_clipps** –

Module contents

wub.mappers package

Submodules

wub.mappers.lastal module

class wub.mappers.lastal.**LastRecord**(*score*, *r_name*, *r_start*, *r_aln_len*, *r_strand*, *r_len*, *r_aln*, *q_name*, *q_start*, *q_aln_len*, *q_strand*, *q_len*, *q_aln*)

Bases: tuple

Create new instance of LastRecord(score, r_name, r_start, r_aln_len, r_strand, r_len, r_aln, q_name, q_start, q_aln_len, q_strand, q_len, q_aln)

q_aln
Alias for field number 12

q_aln_len
Alias for field number 9

q_len
Alias for field number 11

q_name
Alias for field number 7

q_start
Alias for field number 8

q_strand
Alias for field number 10

r_aln
Alias for field number 6

r_aln_len
Alias for field number 3

r_len
Alias for field number 5

r_name
Alias for field number 1

r_start
Alias for field number 2

r_strand
Alias for field number 4

score
Alias for field number 0

wub.mappers.lastal.**check_lastdb_files**(*ref_dir*, *name*)
Check that all lastdb files with *name* label exist within directory

Parameters

- **ref_dir** – directory to check for lastdb files
- **name** – label to search for e.g. 'a' for a.prj

Returns list of missing extensions, [] if none missing

wub.mappers.lastal.**clean_lastdb_files**(*ref_dir*, *name*)
Remove lastdb files having prefix *name* in *ref_dir*.

Parameters

- **ref_dir** – directory to check for lastdb files
- **name** – label to search for e.g. 'a' for a.prj

Returns None

Return type object

wub.mappers.lastal.**compare_genomes_lastal**(*ref_fasta*, *target_fasta*, *filter_alns=True*, *lastal_options=None*, *cleanup=True*)
Compare a refrence set of sequences to a target set os sequences using lastal alignment.

Parameters

- **ref_fasta** – Reference sequence set in fasta format.
- **target_fasta** – Target sequence set in fasta format.
- **filter_alns** – Filter alignments if True.
- **lastal_options** – Options passed to lastal in a dictionary.
- **cleanup** – If True then lastal database files will be deleted.

Returns A pandas data frame with various per-alignment statistics.

Return type DataFrame

wub.mappers.lastal.**filter_top_per_query**(*records*)
Filter lastal alignment records keeping the best scoring one per query.

Parameters **records** – A collection of LastRecord named tuples.

Returns A list of LastRecord named tuples.

Return type list

wub.mappers.lastal.**lastal_align**(*database*, *query*, *executable='lastal'*, ***kwargs*)
Runs lastal via subprocess.

Parameters

- **database** – database prefix
- **query** – filepath for the query file
- **kwargs** – -[args] wanted for lastal e.g. v=" for verbosity

Returns alignment output

wub.mappers.lastal.**lastdb**(*ref_dir*, *ref_name*, *ref*, *executable='lastdb'*, ***kwargs*)
Runs lastdb on ref within ref_dir using the label ref_name if any errors thrown during runtime, files are checked for existence if all files accounted for, successful=False but no errors thrown. Otherwise, IOError or CalledProcessError thrown.

Parameters

- **ref_dir** – directory you will find lastdb files in
- **ref_name** – name of the lastdb files e.g. a for a.prj..
- **ref** – filepath for reference file

- **executable** – path/executable for lastdb e.g. ont_lastdb
- **kwargs** – any -[arg] wanted see lastdb -h for details

Returns True/False is successful with no errors and command run

Raises *IOError* if files don't exist

Raises *subprocess.CalledProcessError* for errors during runtime

wub.mappers.lastal.**parse_lastal**(*res*)
Parse raw lastal output records.

Parameters **res** – Raw lastal results.

Returns Generator of lastal alignment records.

Return type generator

Module contents

wub.parsers package

Submodules

wub.parsers.blastn module

Parser functions for blastn outfmt 6.

wub.parsers.blastn.**parse_coords**(*input_object*)
Parse coordinates file produced by blastn outfmt 6.

Parameters **input_object** – Input path or file hanlder.

Returns List of dictionaries with parsed records.

Return type list

wub.parsers.mummer module

Parser functions for mummer.

wub.parsers.mummer.**parse_coords**(*input_object*)
Parse coordinates file produced by mummer.

Parameters **input_object** – Input path or file hanlder.

Returns List of dictionaries with parsed records.

Return type list

Module contents

wub.read_stats package

Submodules

wub.read_stats.contig_stats module

wub.read_stats.contig_stats.**GC_per_read**(*seq_rec*, *fq=False*)
Calculates the number of bases per sequence, GC content and mean Q score if fastq is given

Parameters

- **seq_rec** – sequence records with attr from biopython
- **fq** – boolean

Returns dataframe

Return type dataframe

wub.read_stats.contig_stats.**L50**(*df*, *col*, *percent=50*)
Calculate the L50 by default however, by changing percent to 75, N75 can be calculated

Parameters

- **df** – dataframe with seqlen column
- **col** – column with sequence length
- **percent** – percentage to be calculated

Returns N50 Value

Return type int

wub.read_stats.contig_stats.**N50**(*df*, *col*, *percent=50*)
Calculate the N50 by default however, by changing percent to 75, N75 can be calculated.

Parameters

- **df** – dataframe with seqlen column
- **col** – column with sequence length
- **percent** – percentage to be calculated

Returns N50 Value

Return type int

wub.read_stats.contig_stats.**get_stats**(*df*)
Calcualtes the summary stats

Parameters **df** – dataframe from GC_per_read

Returns summary Series

Return type Series

wub.read_stats.contig_stats.**readfast**(*fast*)
reads a fasta or fastq file.

Parameters **fast** – fastq or fasta

Returns list of records with attr

Return type generator object

Module contents

wub.simulate package

Submodules

wub.simulate.dist module

wub.simulate.dist.**sample_truncated_gamma**(*mean*, *shape*, *low=None*, *high=None*)
A naive rejection approach to sample from truncated gamma distribution. Note that truncation points ae included in the sample.

Parameters

- **mean** – Mean of the distribution.
- **shape** – Shape parameter.
- **low** – Lower truncation point.
- **high** – Upper truncation point.

Returns Random sample from the specified distribution.

Return type float

wub.simulate.genome module

class wub.simulate.genome.**Fragment**(*chrom*, *uid*, *start*, *end*, *seq*)
Bases: tuple

Create new instance of Fragment(chrom, uid, start, end, seq)

chrom
Alias for field number 0

end
Alias for field number 3

seq
Alias for field number 4

start
Alias for field number 2

uid
Alias for field number 1

wub.simulate.genome.**sample_chromosome**(*chromosomes*)
Sample a random chromosome.

Parameters **chromosomes** – A collection of SeqRecord object.

Returns A randomly sampled element from the input collection.

Return type SeqRecord

wub.simulate.genome.**simulate_fragment**(*chromosome*, *mean_length*, *gamma_shape*, *low_truncation*, *high_truncation*, *fragment_number*)
Simulate a fragment from a chromosome.

Parameters

- **chromosome** – Chromosome to simulate fragment from, SeqRecord object.
- **mean_length** – Mean length of simulated fragment.
- **gamma_shape** – Shape parameter of length distribution.
- **low_truncation** – Minimum read length.
- **high_truncation** – Maximum read length.
- **fragment_number** – The unique identifier of fragment in simulation (number of fragment).

Returns A named tuple with chromosome id, fragment number, start, end and sequence.

Return type namedtuple

wub.simulate.genome.**simulate_fragments**(*chromosomes*, *mean_length*, *gamma_shape*, *low_truncation*, *high_truncation*, *number_fragments*)

Simulate a fragments from a set of chromosomes. Chromosomes are picked randomly for each fragment.

Parameters

- **chromosomes** – Chromosomes to simulate fragment from, a list of SeqRecord objects.
- **mean_length** – Mean length of simulated fragments.
- **gamma_shape** – Shape parameter of length distribution.
- **low_truncation** – Minimum read length.
- **high_truncation** – Maximum read length.
- **number_fragments** – Number of fragments to simulate.

Returns An iterator named tuples with chromosome id, fragment number, start, end and sequence.

Return type generator

wub.simulate.genome.**simulate_genome**(*number_chromosomes*, *mean_length*, *gamma_shape*, *low_truncation*, *high_truncation*, *base_frequencies*)

Generator function for simulating chromosomes in a genome. Chromosome lengths are sampled from a truncated gamma distribution.

Parameters

- **number_chromosomes** – Number of simulated chromosomes.
- **mean_length** – Mean length of simulated chromosomes.
- **gamma_shape** – Shape parameter of the chromosome length distribution.
- **low_truncation** – Minimum chromosome length.
- **high_truncation** – Maximum chromosome length.
- **base_frequencies** – Array of base frequencies in the ACGT order.

Returns A generator of SeqRecord objects.

Return type generator

wub.simulate.seq module

class wub.simulate.seq.**MutatedSeq**(*seq*, *real_qual*, *real_subst*, *real_del*, *real_ins*, *cigar*)
Bases: tuple

Create new instance of MutatedSeq(seq, real_qual, real_subst, real_del, real_ins, cigar)

cigar
Alias for field number 5

real_del
Alias for field number 3

real_ins
Alias for field number 4

real_qual
Alias for field number 1

real_subst
Alias for field number 2

seq
Alias for field number 0

wub.simulate.seq.**add_errors**(*seq*, *nr_errors*, *error_type*)
Introduce a specified number of errors in the target sequence at random positions.

Parameters

- **seq** – Input DNA sequence.
- **nr_errors** – Number of mismatches to introduce.

Returns Mutated sequence.

Return type str

wub.simulate.seq.**cigar_list_to_string**(*cigar_list*)
Sample error type from error weights dictionary.

Parameters **error_weights** – A dcitionary with (type, probability) pairs.

Returns Error type

Return type str

wub.simulate.seq.**compress_raw_cigar_list**(*raw_cigar*)
Sample error type from error weights dictionary.

Parameters **error_weights** – A dcitionary with (type, probability) pairs.

Returns Error type

Return type str

wub.simulate.seq.**random_base**(*probs=[0.25, 0.25, 0.25, 0.25]*)
Generate a random DNA base.

Parameters **probs** – Probabilities of sampling a base, in the ACGT order.

Returns A sampled base.

Return type str

wub.simulate.seq.**random_base_except**(*excluded, probs=[0.25, 0.25, 0.25, 0.25]*)
Generate a random base according to the specified probabilities with the exclusion of the specified base.

Parameters

- **excluded** – Exclude this base from sampling.
- **probs** – Base sampling probabilities in the ACGT order.

Returns A sampled base.

Return type str

wub.simulate.seq.**sample_direction**(*forward_prob*)

wub.simulate.seq.**sample_error_type**(*error_weights*)
Sample error type from error weights dictionary.

Parameters **error_weights** – A dcitionary with (type, probability) pairs.

Returns Error type

Return type str

wub.simulate.seq.**simulate_sequence**(*length, probs=[0.25, 0.25, 0.25, 0.25]*)
Simulate sequence of specified length and base composition.

Parameters

- **length** – Length of simulated sequence.
- **probs** – Base composition vector in the ACGT order.

Returns Simulated sequence.

Return type str

wub.simulate.seq.**simulate_sequencing_errors**(*sequence*, *error_rate*, *error_weights*)
Simulate substitutions, deletions and insertions.

Parameters

- **sequence** – Input sequence.
- **error_rate** – Total error rate.
- **error_weights** – A dictionary with error types as keys and probabilities as values.

The possible error types are: substitution, deletion, insertion. :returns: A named tuple with elements: mutated sequence, realised quality, number of realised substitutions, number of realised deletions, number of realised insertions, cigar string. :rtype: namedtuple

Module contents

wub.tests package

Submodules

wub.tests.test_bam_compare module

class wub.tests.test_bam_compare.**TestBamCompare**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Test BAM comparison test.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

test_bam_read_counter()
Test read_counter wrapper.

wub.tests.test_bam_read_counter module

class wub.tests.test_bam_read_counter.**TestBamReadCounter**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Test BAM read counter wrapper.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

test_bam_read_counter()
Test read_counter wrapper.

wub.tests.test_bam_stats module

class wub.tests.test_bam_stats.**TestBamStats**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Test BAM statistics functions.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

test_error_and_read_stats()
Test the gathering of error and read statistics.

test_fragment_stats()
Test the gathering of fragment statistics.

test_pileup_stats()
Test the gathering read statistics.

test_read_stats()
Test the gathering read statistics.

wub.tests.test_blastn_coord_parse module

class wub.tests.test_blastn_coord_parse.**TestBlastnCoordParse**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

test_nucmer_coord_parse()
Test blastn outfmt 6 cooridnate parsing.

wub.tests.test_contig_stats module

class wub.tests.test_contig_stats.**TestContigStats**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Test N50 utility function.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

test_N50()
Test calculation of N50.

wub.tests.test_example module

class wub.tests.test_example.**ExampleTest**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

setUp()
Hook method for setting up the test fixture before exercising it.

tearDown()
Hook method for deconstructing the test fixture after testing it.

test_success()

wub.tests.test_mappers_lastal module

class wub.tests.test_mappers_lastal.**TestMappersLastal**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

setUp()
Hook method for setting up the test fixture before exercising it.

tearDown()
Hook method for deconstructing the test fixture after testing it.

test_lastal_compare_genomes(***kwargs*)

test_parse_lastal_difference()

test_parse_lastal_identical()

test_parse_lastal_zero()

wub.tests.test_nucmer_coord_parse module

class wub.tests.test_nucmer_coord_parse.**TestNucmerCoordParse**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Test numcmer cooridnate parsing.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

test_nucmer_coord_parse()
Test parsing of nucmer coordinate files.

wub.tests.test_simulate_genome module

class wub.tests.test_simulate_genome.**TestSimulateGenome**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Test genome simulation utilities.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

test_simulate_fragment()
Test fragment simulator.

test_simulate_fragment_edge()
Test fragment simulator (edge case).

test_simulate_genome()
Test genome simulator.

wub.tests.test_simulate_seq module

class wub.tests.test_simulate_seq.**TestSimulateSeq**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Test sequence simulation utilities.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

test_add_errors()
Test function adding sequencing errors.

test_cigar_list_to_string()
Test formatting of cigar strings.

test_compress_raw_cigar_list()
Test compression of raw cigar lists.

test_simulate_sequencing_errors()
Test function simulating sequencing errors.

wub.tests.test_util_parse module

class wub.tests.test_util_parse.**TestUtilParse**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Test parsing utilities.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

test_args_string_to_dict()
Test parsing of dictionaries encoded in separated strings.

test_args_string_to_dict_empty()
Test parsing of dictionaries encoded in separated strings (empty input).

test_normalise_array()
Test array normalization.

test_separated_list_to_floats()
Test parsing of separated lists.

wub.tests.test_util_seq module

class wub.tests.test_util_seq.**TestUtilSeq**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Test sequence utilities.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

test_alignment_stats()
Test calculation of alignment statistics.

test_mean_qscore()
Test mean q score calculation.

test_mean_qscore_large()
Test mean q score calculation (large identical input).

test_mock_qualities()
Test quality mocking function.

test_new_dna_record()
Test the construction of new DNA SeqRecord.

test_phred_to_prob()
Test error probability to phred score conversion.

test_prob_to_phred()
Test error probability to phred score conversion.

test_prob_to_phred_max()
Test error probability to phred score conversion (very small error).

test_reverse_complement()
Test reverse complementing.

wub.tests.test_wrappers_dnadiff module

class wub.tests.test_wrappers_dnadiff.**TestWrappersDnadiff**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Test dnadiff wrapper.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

test_dnadiff(***kwargs*)
Test dnadiff wrapper.

Module contents

wub.util package

Submodules

wub.util.cmd module

Utilities related to running external commands.

wub.util.cmd.**ensure_executable**(*command*)
Find executable in path corresponding to a command and abort if not found.

Parameters **command** – Command.

Returns None

Return type object

wub.util.cmd.**find_executable**(*command*)
Find executable in path corresponding to a command.

Parameters **command** – Command.

Returns Path to executable of False.

Return type str

wub.util.misc module

Yet uncategorised utility functions.

wub.util.misc.**get_extension**(*fname*)
get the file extension.

Parameters **fname** – file name

Returns file extention

Return type str format '.*'

wub.util.misc.**get_fname**(*fname*)
get the file name without extension.

Parameters **fname** – file name

Returns file name

Return type str

wub.util.misc.**mkdir**(*path*)
if the dir does not exists it create it

Parameters **path** – dir path

Returns path

Return type str

wub.util.misc.**pickle_dump**(*obj*, *fname*)
Pickle object to file.

Parameters **obj** – Object to be pickled.

Fname Output file name.

Returns The name of output file.

Return type str

wub.util.misc.**pickle_load**(*fname*)
Load object from pickle.

Parameters **fname** – Input pickle file name.

Returns Object loaded from pickle file.

Return type object

wub.util.parse module

wub.util.parse.**args_string_to_dict**(*args_string*, *elements_separator=', '*, *keyvalue_separator=':'*)
Convert a two-level separated list into a dictionary.

Parameters

- **args_string** – Two-level separated string.
- **elements_separator** – Separator between elements.
- **keyvalue_separator** – Separator between key/value pairs.

Returns dict

Return type dict

wub.util.parse.**interval_string_to_tuples**(*interval_string*, *elements_separator='|'*, *interval_separator=', '*)
Convert a two-level separated list into a dictionary.

Parameters

- **interval_string** – Two-level separated string.
- **elements_separator** – Separator between elements.
- **keyvalue_separator** – Separator between interval boundaries.

Returns tuple

Return type tuple

wub.util.parse.**normalise_array**(*array*)
Normalise numpy array so the elments sum to 1.0.

Parameters **array** – Input array.

Returns Normalised array.

Return type numpy.array

wub.util.parse.**separated_list_to_floats**(*separated_list*, *separator=', '*)
Convert a separated list into a list of floats.

Parameters

- **separated_list** – A separated list as string.
- **separator** – List separator.

Returns List of floats.

Return type list

wub.util.seq module

wub.util.seq.**alignment_stats**(*ref*, *query*, *gap_character='-'*)
Calculate statistics from two aligned sequences.

Parameters

- **ref** – Reference sequence.
- **query** – Query sequence.
- **gap_character** – Gap symbol.

Returns AlnStats namedtuple.

Return type namedtuple

wub.util.seq.**base_complement**(*k*)
Return complement of base.

Performs the subsitutions: A<=>T, C<=>G, X=>X for both upper and lower case. The return value is identical to the argument for all other values.

Parameters **k** – A base.

Returns Complement of base.

Return type str

wub.util.seq.**base_composition**(*seq*)
Return letter counts of a string (base) sequence.

Parameters **seq** – Input sequence.

Returns Letter counts.

Return type dict

wub.util.seq.**count_records**(*input_object*, *format='fasta'*)
Count SeqRecord objects from a file in the specified format.

Parameters

- **input_object** – A file object or a file name.
- **format** – Input format (fasta by default).

Returns Number of records in input file.

Return type int

wub.util.seq.**dna_record_to_rna**(*record*)
Convert a DNA SeqRecord into RNA SeqRecord.

Parameters **record** – DNA SeqRecord.

Returns The RNA SeqRecord object.

Return type SeqRecord

wub.util.seq.**gc_content**(*seq*)
Return fraction of GC bases in sequence.

Parameters **seq** – Input sequence.

Returns GC content.

Return type float

wub.util.seq.**mean_qscore**(*scores*, *qround=True*)
Returns the phred score corresponding to the mean of the probabilities associated with the phred scores provided.

Parameters

- **scores** – Iterable of phred scores.
- **qround** – Round after calculating mean score.

Returns Phred score corresponding to the average error rate, as estimated from the input phred scores.

wub.util.seq.**mock_qualities**(*record*, *mock_qual*)
Add mock quality values to SeqRecord object.

Parameters

- **record** – A SeqRecord object.
- **mock_qual** – Mock quality value used for each base.

Returns The record augmented with mock quality values.

Return type object

wub.util.seq.**new_dna_record**(*sequence*, *name*, *qualities=None*)
Create a new SeqRecord object using IUPACUnambiguousDNA and the specified sequence.

Parameters

- **sequence** – The sequence.
- **name** – Record identifier.
- **qualities** – List of base qualities.

Returns The SeqRecord object.

Return type SeqRecord

wub.util.seq.**phred_to_prob**(*phred*)
Convert phred score into error probability.

Parameters **phred** – Phred quality score.

Returns Error probability.

Return type float

wub.util.seq.**prob_to_phred**(*error_prob*, *max_q=93*, *qround=True*)
Convert error probability into phred score.

Parameters

- **error_prob** – Base error probability.
- **max_q** – Maximum quality value.
- **qround** – Round calculated score.

Returns Phred score.

Return type int

wub.util.seq.**quality_array_to_string**(*quality_list*)
Convert list of phred quality values to string.

Parameters **quality_list** – List of phred quality scores.

Returns Quality string.

Return type str

wub.util.seq.**quality_string_to_array**(*quality_string*)
Convert quality string into a list of phred scores.

Parameters **quality_string** – Quality string.

Returns Array of scores.

Return type array

wub.util.seq.**read_alignment**(*input_file*, *format='fasta'*)
Load multiple alignment from file.

Parameters **input_file** – Input file name.

Returns The alignment read from the input file.

Return type MultipleSeqAlignment

wub.util.seq.**read_seq_records**(*input_object*, *format='fasta'*)
Read SeqRecord objects from a file in the specified format.

Parameters

- **input_object** – A file object or a file name.
- **format** – Input format (fasta by default).

Returns A dictionary with the parsed SeqRecord objects.

Return type generator

wub.util.seq.**read_seq_records_dict**(*input_object*, *format='fasta'*)
Read SeqRecord objects to a dictionary from a file in the specified format.

Parameters

- **input_object** – A file object or a file name.
- **format** – Input format (fasta by default).

Returns An iterator of SeqRecord objects.

Return type dict

wub.util.seq.**record_lengths**(*input_iter*)
Return lengths of SeqRecord obejcts in the input iterator.

Parameters **input_iter** – An iterator of SeqRecord objects.

Returns An ordered dictionary with the lengths of the SeqRecord objects.

Return type OrderedDict

wub.util.seq.**reverse_complement**(*seq*)
Return reverse complement of a string (base) sequence.

Parameters **seq** – Input sequence.

Returns Reverse complement of input sequence.

Return type str

wub.util.seq.**rna_record_to_dna**(*record*)
Convert an RNA SeqRecord into DNA SeqRecord.

Parameters **record** – RNA SeqRecord.

Returns The DNA SeqRecord object.

Return type SeqRecord

wub.util.seq.**word_composition**(*seq*, *size*)
Return word counts of a nucleotide sequence.

Parameters

- **seq** – Input sequence.
- **size** – word length.

Returns word counts.

Return type OrderedDict

wub.util.seq.**write_seq_records**(*records_iterator*, *output_object*, *format='fasta'*)
Write out SeqRecord objects to a file from an iterator in the specified format.

Parameters

- **records_iterator** – An iterator of SeqRecord objects.
- **output_object** – Open file object or file name.
- **format** – Output format (fasta by default).

Returns None

Return type object

Module contents

wub.vis package

Submodules

wub.vis.report module

class wub.vis.report.**Report**(*pdf*)
Class for plotting utilities on the top of matplotlib. Plots are saved in the specified file through the PDF backend.

Parameters

- **self** – object.
- **pdf** – Output pdf.

Returns The report object.

Return type *Report*

close()
Close PDF backend. Do not forget to call this at the end of your script or your output will be damaged!

Parameters **self** – object

Returns None

Return type object

plot_arrays(*data_map*, *title=''*, *xlab=''*, *ylab=''*, *marker='.'*, *legend_loc='best'*, *legend=True*, *vlines=None*, *vlcolor='green'*, *vlwitdh=0.5*)

Plot multiple pairs of data arrays.

Parameters

- **self** – object.
- **data_map** – A dictionary with labels as keys and tupples of data arrays (x,y) as values.
- **title** – Figure title.
- **xlab** – X axis label.
- **ylab** – Y axis label.
- **marker** – Marker passed to the plot function.
- **legend_loc** – Location of legend.
- **legend** – Plot legend if True
- **vlines** – Dictionary with labels and positions of vertical lines to draw.
- **vlcolor** – Color of vertical lines drawn.
- **vlwidth** – Width of vertical lines drawn.

Returns None

Return type object

plot_bars_simple(*data_map*, *title=''*, *xlab=''*, *ylab=''*, *alpha=0.6*, *xticks_rotation=0*, *auto_limit=False*)

Plot simple bar chart from input dictionary.

Parameters

- **self** – object.
- **data_map** – A dictionary with labels as keys and data as values.
- **title** – Figure title.
- **xlab** – X axis label.
- **ylab** – Y axis label.
- **alpha** – Alpha value.
- **xticks_rotation** – Rotation value for x tick labels.
- **auto_limit** – Set y axis limits automatically.

Returns None

Return type object

plot_boxplots(*data_map*, *title=''*, *xlab=''*, *ylab=''*, *xticks_rotation=0*, *xticks_fontsize=5*)

Plot multiple pairs of data arrays.

Parameters

- **self** – object.
- **data_map** – A dictionary with labels as keys and lists as data values.
- **title** – Figure title.
- **xlab** – X axis label.

- **ylab** – Y axis label.
- **xticks_rotation** – Rotation value for x tick labels.
- **xticks_fontsize** – Fontsize for x tick labels.

Returns None

Return type object

plot_dicts(*data_map*, *title=''*, *xlab=''*, *ylab=''*, *marker='-'*, *legend_loc='best'*, *legend=True*, *hist_style=False*, *cmap=<matplotlib.colors.LinearSegmentedColormap object>*, *alpha=0.6*)

Plot elements of multiple dictionaries on a single plot.

Parameters

- **self** – object.
- **data_map** – A dictionary with labels as keys and dictionaries as values.
- **title** – Figure title.
- **xlab** – X axis label.
- **ylab** – Y axis label.
- **marker** – Marker passed to the plot function.
- **legend_loc** – Location of legend.
- **legend** – Hide legend if False.
- **hist_style** – Plot histogram-style bar plots.
- **cmap** – Colormap for histogram plots.
- **alpha** – Transparency value for histograms.

Returns None

Return type object

plot_heatmap(*data_matrix*, *title=''*, *xlab=''*, *ylab=''*, *colormap=<matplotlib.colors.LinearSegmentedColormap object>*)

Plot heatmap of data matrix.

Parameters

- **self** – object.
- **data_matrix** – 2D array to be plotted.
- **title** – Figure title.
- **xlab** – X axis label.
- **ylab** – Y axis label.
- **colormap** – matplotlib color map.

Retuns None

Return type object

plot_histograms(*data_map*, *title=''*, *xlab=''*, *ylab=''*, *bins=50*, *alpha=0.7*, *legend_loc='best'*, *legend=True*, *vlines=None*)

Plot histograms of multiple data arrays.

Parameters

- **self** – object.
- **data_map** – A dictionary with labels as keys and data arrays as values.
- **title** – Figure title.
- **xlab** – X axis label.
- **ylab** – Y axis label.
- **bins** – Number of bins.
- **alpha** – Transparency value for histograms.
- **legend_loc** – Location of legend.
- **legend** – Plot legend if True.
- **vlines** – Dictionary with labels and positions of vertical lines to draw.

Returns None

Return type object

plot_line(*data*, *x*, *y*, *title=''*, *xlab=''*, *ylab=''*)

Generate a line plot from pandas dataframe

Parameters

- **data** – pandas dataframe
- **x** – X axis data
- **y** – Y axis data
- **title** – Figure title
- **xlab** – X axis label
- **ylab** – Y axis label

Returns None

Return type object

plot_pcolor(*data*, *title=''*, *xlab=''*, *ylab=''*, *xticks=None*, *yticks=None*, *invert_yaxis=False*, *colormap=<matplotlib.colors.LinearSegmentedColormap object>*, *tick_size=5*, *tick_rotation=90*)

Plot square heatmap of data matrix.

Parameters

- **self** – object.
- **data** – 2D array to be plotted.
- **title** – Figure title.
- **xlab** – X axis label.
- **ylab** – Y axis label.
- **xticks** – X axis tick labels..
- **yticks** – Y axis tick labels..
- **invert_yaxis** – Invert Y axis if true.
- **colormap** – matplotlib color map.
- **tick_size** – Font size on tick labels.

- **tick_rotation** – Rotation of tick labels.

Retuns None

Return type object

plot_scatter(*data*, *x*, *y*, *title=''*, *xlab=''*, *ylab=''*, *alpha=0.5*, *ylim=None*, *xlim=None*)
Generates a scatter plot from a pandas dataframe

Parameters

- **data** – Pandas dataframe
- **x** – X axis data
- **y** – Y axis data
- **title** – Figure title
- **xlab** – X axis label
- **ylab** – Y axis label
- **alpha** – opacity of data pionts
- **ylim** – Y axis limit
- **xlim** – X axis limit

Returns None

Return type object

Module contents

wub.wrappers package

Submodules

wub.wrappers.dnadiff module

Wrapper for mummer's dnadiff

class wub.wrappers.dnadiff.**Property**(*ref*, *query*)
Bases: tuple

Create new instance of Property(ref, query)

query
Alias for field number 1

ref
Alias for field number 0

class wub.wrappers.dnadiff.**PropertyWithPerc**(*ref*, *ref_perc*, *query*, *query_perc*)
Bases: tuple

Create new instance of PropertyWithPerc(ref, ref_perc, query, query_perc)

query
Alias for field number 2

query_perc
Alias for field number 3

ref
Alias for field number 0

ref_perc
Alias for field number 1

wub.wrappers.dnadiff.**cleanup_dnadiff_report**(*directory*, *prefix='out'*)
Cleanup dnadiff output files in the specified directory.

Parameters

- **directory** – Output directory.
- **prefix** – Output prefix.

Returns None

Return type object

wub.wrappers.dnadiff.**dnadiff**(*reference*, *query*, *working_directory=None*, *cleanup=True*)
Run dnadiff on reference and query fasta and parse results.

Parameters

- **reference** – Reference fasta.
- **query** – Query fasta.
- **working_directory** – Write output in this directory if specified.
- **cleanup** – Delete dnadiff output after parsing if True.

Returns Parsed results, raw report and log.

Return type 3-tuple

wub.wrappers.dnadiff.**parse_dnadiff_report**(*report_file*)
Parse dnadiff report file.

Parameters **report_file** – dnadiff report output.

Returns Data structure with parsed results.

Return type dict

Module contents

Module contents

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

W

Index

I

L

M

N

P

Q

R

S

T

U

W