User Manual

Content

0. Download and installation

SLALOM is an open-source software and can be freely downloaded from the GitHub repository: https://github.com/BCF-calanques/SLALOM. From a Unix terminal:

```
$git clone https://github.com/BCF-calanques/SLALOM
```

This will generate the directory `SLALOM/` with the code in the subdirectory `SLALOM/src/` and all the mentioned test files in the subdirectory `SLALOM/test_data/`.

SLALOM is distributed as a collection of Python scripts. Therefore, it is cross-platform and does not require additional installation. SLALOM is a command-line tool and requires a Python interpreter of version 3.5 or later. The latest version of Python for different platforms can be obtained from https://www.python.org/downloads/.

In addition to Python, SLALOM depends on the Python library `numpy`. To install `numpy` from the command line:

```
$pip install numpy
```

That's all! There are no further dependencies and SLALOM can be used from a terminal by providing the path to slalom.py. For example:

```
$python SLALOM/src/slalom.py --help
```

1. The input files

SLALOM takes four input files: two annotations, sequence length database, and group mapping, the last two being optional.

The main input files are the two annotations to compare. These files are mandatory for every mode SLALOM can be started in. In the basic case, they contain the sequence identifier (SID) the start and the end positions of the continuous sequence elements (CSEs; a.k.a. sites, ranges or intervals), for example:

```
$cat test_anno1.tsv
seqA  6     10
seqA  8     12
seqA  9     18
seqB  15    25
$cat test_anno2.tsv
```

```
seqA  5     11
seqA  7     10
seqA  9     15
seqB  15    19
seqB  21    25
```

By default, the numeration of positions (a.k.a. symbols or residues) starts from 1. Both start and end positions are considered included. That means that the length of a CSE with the start at 15 and the end at 25 is 11 symbols. The CSEs can overlap, contain or duplicate each other.

The annotation files must be provided with the keys `-a1/--anno1file` and `-a2/--anno2file`.

To complete the command line input, the user needs to provide the sequence length (say, both `seqA` and `seqB` are 50 symbols long) with the key `-l/--seqlen_value` and specify the output file with the key `-o/--outfile`:

```
$python slalom.py -a1 test_anno1.tsv -a2 test_anno2.tsv -l 50 -o output.tsv
```

Alternatively, the sequence length can be provided in the sequence length database file, which contains the same SIDs and corresponding integer values:

```
$cat test_seqlenfile1.tsv
seqA  50
seqB  30
seqC  45
seqD  60
```

The sequence length database file is the only option, when there is more than one sequence and they have different lengths. For convenience, it can also contain SIDs not mentioned in the annotations (`seqC` and `seqD` in this example). The file must be provided with the key `-s/--seqlenfile`:

```
$python slalom.py -a1 test_anno1.tsv -a2 test_anno2.tsv -s test_seqlenfile1.tsv -o
output.tsv
```

2. Analysis of grouped sequences

Sometimes the sequences are grouped into categories and the analysis of each category separately is desired. If this is the case, then the group mapping can be provided as separate file containing SIDs mapped to group identifiers (GIDs) in the long format:

```
$cat test_groupmapping.tsv
seqA  group0
seqB  group0
seqB  group1
seqC  group1
seqD  group2
```

The groups can contain different numbers of sequences and be overlapping (`seqB` in the provided example belongs to both `group0` and `group1`). For this reason, the annotation files must be adjusted accordingly. Now, a CSE can belong to a certain sequence in one group but not in the other. To reflect this, another column must be added to the annotation files:

```
$cat test_anno3.tsv
seqA  group0      8     15
seqA  group0      21    25
seqB  group0      6     9
seqB  group1      8     11
seqC  group1      18    24
seqD  group2      1     9
```

```
seqD  group2      5     12
$cat test_anno4.tsv
seqB  group1      8     12
seqC  group1      11    16
seqD  group2      5     10
```

In this example, the two SCEs in `seqB` in the first annotation are no longer considered overlapping for the analysis, as they occur in different groups.

The group-mapping file must be provided with the key `-m/--mapfile`:

```
$python slalom.py -a1 test_anno3.tsv -a2 test_anno4.tsv -s test_seqlenfile1.tsv -m
test_groupmapping.tsv -o output.tsv
```

In the provided example, the first annotation already includes the group mapping (duplicating records are no problem). Therefore, the same file can be used for both inputs:

```
$python slalom.py -a1 test_anno3.tsv -a2 test_anno4.tsv -s test_seqlenfile1.tsv -m
test_anno3.tsv -o output.tsv
```

This framework of four input files is designed to provide flexibility in general cases. However, in many real situations, there is some additional information available allowing for simpler input. For example, sometimes it is known that the groups are not overlapping, i.e., that each sequence belongs to only one group. In this case, the GIDs are no longer necessary in the annotation files, which can be indicated by providing the flag `-nOg/--non_overlapping_groups`. Furthermore, if each group consists only of one sequence, the user can activate the option of treating SIDs as GIDs, without supplying the group-mapping file, by providing the flag `-sg/--sequences_as_groups`. These and other flags useful in simplified cases are discussed in more detail further down in this manual.

3. The main output file

The main output file begins with several commented (i.e., starting with `#`) lines, containing information about the input command line as well as about the statistics calculated:

```
# This file was generated at 2017-01-01 00:00:01 with SLALOM
# Command line options (unquoted and unescaped): -a1 test_anno3.tsv -a2 test_anno4.tsv
-s test_seqlenfile1.tsv -m test_anno3.tsv -o output.tsv
# The following statistics have been calculated:
#    Nseq: Number of sequences
#    P1: Share of symbols present in the first annotation
#    P2: Share of symbols present in the second annotation
#    PP: Share of symbols present in both annotations
#    AP: Share of symbols absent in the first annotation but present in the second
#    PA: Share of symbols present in the first annotation but absent in the second
#    AA: Share of symbols absent in both annotations
#    ACC: Symbol-wise accuracy
#    MCC: Symbol-wise Matthews correlation coefficient
#    F1: Symbol-wise F1 score
#    SiteN1: Number of sites in the first annotation
#    SiteN2: Number of sites in the second annotation
#    SiteN1m: Number of matched sites in the first annotation
#    SiteN2m: Number of matched sites in the second annotation
#    SiteL1: Total length of sites in the first annotation
#    SiteL2: Total length of sites in the second annotation
#    SiteF1: Site-wise F1 score
#    SitePCV: Site-wise positive correlation value
# All the provided dataset-wide averages are macro-averages of the group-wide metrics
```

This block is followed by the table with the first row being header containing the column names. The commented block is helpful at the first usages of SLALOM. However, for easier further processing of the output, it can be suppressed with the flag `-c/--clean`:

```
$python slalom.py -a1 test_anno3.tsv -a2 test_anno4.tsv -s test_seqlenfile1.tsv -m
test_groupmapping.tsv -o output.tsv -c
```

If this flag is provided, only the actual table (with the header row is printed):

```
$cat output.tsv
Group   Nseq    P1      …  SiteN1  SiteN2  SiteN1m SiteN2m SiteL1  SiteL2  …
group0  2       0.2125  …  3       0       0       0       17      0       …
group1  2       0.1467  …  2       2       1       1       11      11      …
group2  1       0.2000  …  2       1       2       1       12      6       …
Average 1.6667  0.1864  …  2.3333  1.0000  1.0000  0.6667  13.333  5.6667  …
```

For metrics like total site length the sum may be more informative than average. The flag `-sum/--calculate_sums` will add the corresponding row to the output:

```
$python slalom.py -a1 test_anno3.tsv -a2 test_anno4.tsv -s test_seqlenfile1.tsv -m
test_groupmapping.tsv -o output.tsv –c -sum --quiet
$cat output.tsv
Group   Nseq    P1      …  SiteN1  SiteN2  SiteN1m SiteN2m SiteL1  SiteL2  …
group0  2       0.1967  …  3       0       0       0       17      0       …
group1  2       0.1444  …  2       2       1       1       11      11      …
group2  1       0.2000  …  2       1       2       1       12      6       …
Average 1.6667  0.1804  …  2.3333  1.0000  1.0000  0.6667  13.333  5.6667  …
Sum     5               …  7       3       3       2       40      17      …
```

Note that sequences belonging to multiple groups are counted the corresponding numbers of times.

4. The operating modes

SLALOM supports three different approaches of handling overlapping CSEs in one annotation. In addition, the annotations can be treated symmetrically – when they have similar origins and reliability – or the first annotation can be used as benchmark to evaluate the second one. Together, this results six operating modes. The operating modes should not be confused with the simplified modes for easier processing of certain file formats and can be used independently of those.

In the output file shown in the previous section, one can see that the total length of CSEs in the first annotation (the column `SiteL1`) for `group2` equals to 12 or 20% share (the column `P1`). This means that implicit merging of the two overlapping CSEs (`1`-`9` and `5`-`12`) happened. Note, however, that the number of sites in the first annotation (the column `SiteN1`) still equals to 2 and the site-based statistics are calculated accordingly. This behavior is termed ‘symbol-resolved mode’ and is applied by default.

Sometimes, however, there is a need to count the symbols as many times as they occur in a given annotation, i.e., traversed by a CSE. This behavior is termed ‘gross mode’ and is invoked by providing the value `gross` with the key `-E/--enrichment_count`:

```
$python slalom.py -a1 test_anno3.tsv -a2 test_anno4.tsv -s test_seqlenfile1.tsv -m
test_groupmapping.tsv -o output.tsv -E gross -c --quiet
$cat output.tsv
Group   Nseq    …  SiteN1  SiteN2  SiteN1m SiteN2m SiteL1  SiteL2  …
group0  2       …  3       0       0       0       17      0       …
group1  2       …  2       2       1       1       11      11      …
group2  1       …  2       1       2       1       17      6       …
Average 1.6667  …  2.3333  1.0000  1.0000  0.6667  15.000  5.6667  …
```

This time, the total length of CSEs in the first annotation is 17; the share of symbols is not calculated in the gross mode, as the metric could exceed 1 now.

In some other situations, the user may want to consider only those positions that are traversed by certain minimal number of CSEs – the 'enrichment mode'. This number must be provided with the key `-E/--enrichment_count`. In the enrichment mode, the calculated metrics have slightly different names. For example, the share of symbols *present* in the first annotation (the column `P1`) becomes the share of symbols *enriched* (the column `E1`). If it is set, for example, to 2, this results in considering only those positions that belong at least to 2 CSEs. In the provided example, there are no such positions in the first annotation for group0 and group1; for group2, there are only 5 positions (5-9 in `seqD`) for 60 symbols, which results in the share of 0.0833:

```
$python slalom.py -a1 test_anno3.tsv -a2 test_anno4.tsv -s test_seqlenfile1.tsv -m
test_groupmapping.tsv -o output.tsv -E 2 -c --quiet
$cat output.tsv
Group   Nseq    E1      …
group0  2       0.0000  …
group1  2       0.0000  …
group2  1       0.0833  …
Average 1.6667  0.0278  …
```

In addition to different strategies of resolving the overlaps within one annotation, the annotations can be in different relation to each other. One scenario is comparing annotations, which relate to the same type of CSEs (e.g., genes and genes), come from similar origins, and have comparable quality. This scenario is called 'equal mode' and is assumed by default. However, when CSEs in different annotations represent different types of regions (e.g., genes and promoters) or have significantly different reliability (e.g., experimentally verified and predicted), a different scenario is in play. This scenario is called 'benchmarking mode' and is assumed if the flag `-b/--benchmarking` is provided:

```
$python slalom.py -a1 test_anno3.tsv -a2 test_anno4.tsv -s test_seqlenfile1.tsv -m
test_groupmapping.tsv -o output.tsv -b -c
```

The change from the equal to benchmarking mode involves both naming conventions as well as enabling calculation of additional performance measures. The names are chosen to highlight the non-symmetry in the benchmarking mode: the 'first annotation' becomes 'benchmark', the 'second annotation' becomes 'prediction', the metric 'present in both annotations' becomes 'true positive', etc. Nevertheless, changes in names do not reflect on the calculations: the values of the outputted metrics are equivalent in the both modes. In the equal mode, however, only symmetric performance measures – those unaffected by flipping the annotation files – are calculated. Accuracy (ACC) and F1 score are examples of symmetric measures. In the benchmarking mode, additional non-symmetric measures are calculated, for example, specificity (SPC) and precision (PPV). Thus, the equal mode is effectively just a subset of the benchmarking mode and brings only small speed and the output file size advantages; however, it was implemented to prevent the user from misinterpreting certain metrics in the case of symmetric annotations.

5. Annotations in GenBank format

Although SLALOM works primarily with tabular input, it has a special simplified mode to compare genome annotations in GenBank format. This mode works under assumption that the both annotations are of the same sequence. The sequence length is automatically read from the files and must not be provided separately. To start the program in the GenBank mode, the user needs to provide the `--genbank` flag:

```
$python slalom.py --genbank -a1 N_pharaonis_annotation1.gb -a2
N_pharaonis_annotation2.gb -o comparison_stats.tsv -c
```

In the given example, there will be one data row in the table, as only one sequence (although perhaps a very long one) is analyzed:

```
$cat comparison_stats.tsv
Nseq    …  SiteN1  SiteN2  SiteN1m SiteN2m SiteL1  SiteL2  SiteF1  SitePCV
1       …  2694    2608    2622    2602    2359033 2336204 0.9853  0.9853
```

However, the user can analyze each strand or reading frame separately. For this, the value `strand` or `frame` respectively must be provided with the key `-d/--detect`:

```
$python slalom.py --genbank -a1 N_pharaonis_annotation1.gb -a2
N_pharaonis_annotation2.gb -o comparison_stats.tsv -c -d frame -sum
```

Now, the output file contains eight rows: the header, statistics for each of the six reading frames, and the bottom lines with the averages and sums :

```
$cat comparison_stats.tsv
Frame   Nseq    …  SiteN1  SiteN2  SiteN1m SiteN2m SiteL1  SiteL2  SiteF1  SitePCV
+1      1       …  475     458     453     453     419436  410952  0.9711  0.9711
+2      1       …  473     460     459     459     416979  411405  0.9839  0.9839
+3      1       …  459     441     439     440     400059  394497  0.9766  0.9767
-1      1       …  456     438     436     436     384372  382003  0.9754  0.9754
-2      1       …  382     376     372     372     344138  339013  0.9815  0.9815
-3      1       …  449     435     430     431     402931  401475  0.9740  0.9740
Average 1.0000  …  449.00  434.67  431.50  431.83  394652  389891  0.9771  0.9771
Sum     6       …  2694    2608    2589    2591    2367915 2339345
```

From this table, one can see, for example, that there are total 475 genes in the reading frame `+1` of the first annotation but only 458 genes for the second annotation. Moreover, 453 genes in each annotation overlap with a gene from another annotation (at this stage, it is not possible to conclude that all the matches are reciprocal). The total length of the genes is 419,436 bp for the first annotation and 410,952 bp for the second annotation. If there are for some reason any overlapping genes annotated, they are merged for this residue calculation, as SLALOM is operating in the default symbol-resolved mode. Also because of the merging, the total gene length without the frame separation (2,359,033 bp) is less than the sum of the frame gene lengths for the first annotation (2,367,915 bp).

6. Annotations in BED format

SLALOM offers a simplified mode to process annotations in BED format. To use it, the user has to provide the flag `--bed`, as well as the two annotation files, the sequence length database, and the output file:

```
$cat test_bed1.bed
chr1    0       10
chr1    15      25
chr1    40      50
chr2    12      27
$cat test_bed2.bed
chr1    0       12
chr1    25      42
$cat test_seqlenfile2.tsv
chr1    50
chr2    40
$python slalom.py --bed -a1 test_bed1.bed -a2 test_bed2.bed -s test_seqlenfile2.tsv -o
output.tsv -c
```

Note that although SLALOM by default counts residues starting from 1 with both start and end residues being included, it is automatically adjusted to the BED format specifications, which require counting starting from 0 with only the start being included.

The default output table contains only the summary line:

```
$cat output.tsv
Nseq    P1      …  SiteN1  SiteN2  SiteN1m SiteN2m SiteL1  SiteL2  SiteF1  SitePCV
2       0.5000  …  4       2       2       2       45      29      0.6667  0.6667
```

As the BED format does not explicitly specify group identifiers, grouping of sequences is not possible in this simplified mode. However, with the flag `-sg/--sequences_as_groups` each sequence will be treated as group on its own and the statistics for each sequence will be printed separately:

```
$python slalom.py --bed -a1 test_bed1.bed -a2 test_bed2.bed -s test_seqlenfile2.tsv -o
output.tsv -c -sg -sum --quiet
$cat output.tsv
Seq.    Nseq    P1      …  SiteN1  SiteN2  SiteN1m SiteN2m SiteL1  SiteL2  SiteF1  …
chr1    1       0.6000  …  3       2       2       2       30      29      0.8000  …
chr2    1       0.3750  …  1       0       0       0       15      0       nan     …
Average 1.0000  0.4875  …  2.0000  1.0000  1.0000  1.0000  22.500  14.500  0.8000  …
Sum     2               …  4       2       2       2       45      29              …
```

If the BED files contain the strand information, the statistics for each strand separately can be calculated:

```
$cat test_bed3.bed
chr1    0       10      featA   0       +
chr1    15      25      featB   0       -
chr1    40      50      featC   0       +
chr2    12      27      featD   0       -
$cat test_bed4.bed
chr1    0       12      featE   0       +
chr1    25      42      featF   0       -
$python slalom.py --bed -a1 test_bed3.bed -a2 test_bed4.bed -s test_seqlenfile2.tsv -o
output.tsv -c -d strand -sum --quiet
$cat output.tsv
Seq.    Nseq    P1      …  SiteN1  SiteN2  SiteN1m SiteN2m SiteL1  SiteL2  SiteF1  …
chr1+   1       0.4000  …  2       1       1       1       20      12      0.6667  …
chr1-   1       0.2000  …  1       1       0       0       10      17      nan     …
chr2-   1       0.3750  …  1       0       0       0       15      0       nan     …
Average 1.0000  0.3250  …  1.3333  0.6667  0.3333  0.3333  15.000  9.6667  0.6667  …
Sum     3               …  4       2       1       1       45      29              …
```

Note that in case of strand or frame detection, the option of treating sequences as groups is activated automatically.

7. Averaging approaches

In the previous two sections, one could have noticed that some metrics – for example, the site-wise F1 score – change their value merely on turning on and off a sequence grouping option. This is not a bug in SLALOM but a consequence of differences between micro and macro averaging. With micro averaging, all the symbol and site counts are summed up before the performance metrics formulas are applied. With macro averaging, the formulas applied first and the results are simple-averaged to produce the displayed metrics. Averaging is applied twice – to the sequence data at the group level as well as to the group data at the dataset level – and is regulated through the key `-a/--averaging`. Providing value `sequence` will trigger macro averaging on both levels, while providing `dataset` will trigger micro averaging; the default value `group` sets micro averaging at the group-level and macro averaging at the dataset level.

Thus, unless dataset-wise averaging is applied, the average value for a column is the simple average of the corresponding group-wide values (`nan`s are ignored):

```
$python slalom.py --bed -a1 test_bed3.bed -a2 test_bed4.bed -s test_seqlenfile2.tsv -o
output.tsv -c -d strand -sum --benchmarking --quiet
$cat output.tsv
Seq.    Nseq    …  SiteNB  SiteNP  SiteNBm SiteNPm SiteLB  SiteLP  SiteTPR SitePPV …
chr1+   1       …  2       1       1       1       20      12      0.5000  1.0000  …
chr1-   1       …  1       1       0       0       10      17      0.0000  0.0000  …
chr2-   1       …  1       0       0       0       15      0       0.0000  nan     …
Average 1.0000  …  1.3333  0.6667  0.3333  0.3333  15.000  9.6667  0.1667  0.5000  …
Sum     3       …  4       2       1       1       45      29                      …
```

In the benchmarking mode, the site-wise TPR (true positive rate; a.k.a. recall or sensitivity) is the ratio of the number of matched benchmark sites (`SiteNBm`) to the total number of benchmark sites (`SiteNB`) and the site-wise PPV (positive predictive value; a.k.a. precision) is the ratio of the number of matched predicted sites (`SiteNPm`) to the total number of predicted sites (`SiteNP`). These numbers are calculated for each group and then averaged.

Under dataset-wise averaging, however, the respective calculations are performed on the summed-up numbers of sites:

```
$python slalom.py --bed -a1 test_bed3.bed -a2 test_bed4.bed -s test_seqlenfile2.tsv -o
output.tsv -c -d strand -sum -b -a dataset --quiet
$cat output.tsv
Seq.    Nseq    …  SiteNB  SiteNP  SiteNBm SiteNPm SiteLB  SiteLP  SiteTPR SitePPV …
chr1+   1       …  2       1       1       1       20      12      0.5000  1.0000  …
chr1-   1       …  1       1       0       0       10      17      0.0000  0.0000  …
chr2-   1       …  1       0       0       0       15      0       0.0000  nan     …
Average 1.0000  …  1.3333  0.6667  0.3333  0.3333  15.000  9.6667  0.2500  0.5000  …
Sum     3       …  4       2       1       1       45      29                      …
```

Only one site out of four was recalled for the whole dataset, and therefore the site-wise TPR is now 0.25. The site-wise PPV remains at 0.5, although on different data it would generally also be different.

Another aspect of averaging is sequence length adjustment. Depending on the underlying question, a site of length 5 in a sequence of length 100 may have different weight than another site of length 5 but in a sequence of length 25. As a result, it may happen that if only one of such two sites in a benchmark annotation was recalled/predicted correctly in another annotation, the user would like to see different values depending on which site exactly is predicted correctly. Under the default settings, the performance metrics are not adjusted for the sequence length, and therefore the results are equivalent for both situations:

```
$cat test_seqlenfile3.tsv
seqA    100
seqB    25
$cat test_anno_benchmark.tsv
seqA    21      25
seqB    6       10
$cat test_anno_prediction1.tsv
seqA    21      25
seqB    16      20
$cat test_anno_prediction2.tsv
seqA    41      45
seqB    6       10
$python slalom.py -a1 test_anno_benchmark.tsv -a2 test_anno_prediction1.tsv -s
test_seqlenfile3.tsv -o output.tsv -c -b --quiet
$cat output.tsv
Nseq    …  TP      FP      FN      …  TPR     PPV     SPC     …  SiteTPR SitePPV …
2       …  0.0400  0.0400  0.0400  …  0.5000  0.5000  0.9565  …  0.5000  0.5000  …
$python slalom.py -a1 test_anno_benchmark.tsv -a2 test_anno_prediction2.tsv -s
test_seqlenfile3.tsv -o output.tsv -c -b --quiet
$cat output.tsv
Nseq    …  TP      FP      FN      …  TPR     PPV     SPC     …  SiteTPR SitePPV …
2       …  0.0400  0.0400  0.0400  …  0.5000  0.5000  0.9565  …  0.5000  0.5000  …
```

The output changes accordingly, if the flag `-A/--adjust_for_seqlen` is provided. It acts like a variation of sequence-wise (macro) averaging at the group level. While by default the sequence lengths and TP (true positive), FP (false positive), etc. counts are summed-up for the whole group and then used to calculate the respective shares (columns `TP`, `FP`, etc.), with the adjustment the shares are calculated for each sequence and then simple-averaged:

```
$python slalom.py -a1 test_anno_benchmark.tsv -a2 test_anno_prediction1.tsv -s
test_seqlenfile3.tsv -o output.tsv -c -b -A --quiet
$cat output.tsv
Nseq    …  TP      FP      FN      …  TPR     PPV     SPC     …  SiteTPR SitePPV …
2       …  0.0250  0.1000  0.1000  …  0.2000  0.2000  0.8857  …  0.5000  0.5000  …
$python slalom.py -a1 test_anno_benchmark.tsv -a2 test_anno_prediction2.tsv -s
test_seqlenfile3.tsv -o output.tsv -c -b -A --quiet
$cat output.tsv
Nseq    …  TP      FP      FN      …  TPR     PPV     SPC     …  SiteTPR SitePPV …
2       …  0.1000  0.0250  0.0250  …  0.8000  0.8000  0.9714  …  0.5000  0.5000  …
```

Note that this adjustment may have quite significant influence on performance metrics, if the sequence lengths are very diverse. However, it affects only the symbol-wise measures and not the site-wise ones. Note also that the adjustment for sequence length differs from sequence-wise averaging:

```
$python slalom.py -a1 test_anno_benchmark.tsv -a2 test_anno_prediction1.tsv -s
test_seqlenfile3.tsv -o output.tsv -c -b -a sequence --quiet
$cat output.tsv
Nseq    …  TP      FP      FN      …  TPR     PPV     SPC     …  SiteTPR SitePPV …
2       …  0.0400  0.0400  0.0400  …  0.5000  0.5000  0.9737  …  0.5000  0.5000  …
```

8. Extended input format options

In the earlier sections of this manual, it was shown that SLALOM can process without further specifications input files in some predefined formats as well as tabular files, which satisfy certain conditions. These conditions are the following:

- The column delimiter is tab
- The columns are following the order SID - GID - start - end - name (non-specified columns omitted) for the annotation files, SID - length for the sequence length database, and SID - GID for the group mapping file
- There are no further columns before or between the specified (additional columns to the right will be ignored)
- There are no header rows
- Quotes are not part of the identifiers and names (although tabs may be if quoted)

Therefore, the user always has an option to convert the input files in one of the supported formats to process them with SLALOM in an easy way. However, this is not always convenient, as there are plenty of tabular data files – available for download or outputted by software tools – that do not strictly conform to requirements of the supported format. Furthermore, the user may encounter situations, when the annotations come from different sources and therefore have different formats. To offer flexibility also for these cases, SLALOM has extended options allowing it to read virtually any kind of tabular data files.

The format options are combinations of prefixes and suffixes. The prefixes correspond to the file name keys: `-a1` for the first/benchmark annotation, etc. The suffixes are given in the following table:

Short version	Long version	Description
`d`	`delim`	Specifies the column delimiter (the following delimiters are supported: tab, space, comma, dot, semicolon, colon, slash)

c	colnumbers	Specifies the column numbers in the following order: start - end - SID - GID - name for the annotation files; SID - length (SID - start - end for time series) for the sequence length database; SID - GID for the group mapping
h	headers	Specifies the number of header rows at the beginning of the file
q	quotes	Triggers the literal treatment of quotes (quoted delimiters are treated as delimiters; single – but not double – quotes can be part of identifiers and names)

A prefix is combined with a suffix to form one of the 16 possible option keys and flags. For the long options, the extra underscore is added as separator. Therefore, to specify, for example, that the second annotation file has three header rows that need to be skipped, the user should provide the value `3` with the key `-a2h/--anno2file_headers`. The column numbers are integers, so that the leftmost column has the number 1. They must be provided comma-separated without additional spaces. For example, if the sequence lengths are read from a file that contains the identifiers in the fifth column and the length values in the third column, the user should provide the value `5,3` with the key `-sc/--seqlenfile_colnumbers`.

This approach allows that same files and/or same columns may be used several times if the input format requires so. In the following example, the sequence lengths and the group mapping are read from the annotation files:

```
$cat test_anno5.txt
Number;ID;Quality;Score;Start;End
1;ABC;Good;0.76;17;55
2;ABC;Good;1.02;58;70
3;DEF;Good;1.21;12;28
4;XYZ;Bad;0.18;101;108
$cat test_anno6.csv
#This file was generated by …
Origin,Sequence,Length,Peak Value,Average value,Start,End
Natural,ABC,100,8.4,2.0,48,62
Artif.,DEF,120,9.1,1.9,26,40
Artif.,DEF,120,6.2,0.9,69,85
Artif.,XYZ,150,11.5,3.1,108,122
$python slalom.py -a1 test_anno5.txt -a1h 1 -a1c 5,6,2 -a1d ";" -a2 test_anno6.csv -a2h
2 -a2c 6,7,2 -a2d "," -s test_anno6.csv -sh 2 -sc 2,3 -sd "," -m test_anno5.txt -mh 1 -
mc 2,3 -md ";" -o output.tsv -nOg -c --quiet
$cat output.tsv
Group   Nseq    P1      P2      …  SiteN1  SiteN2  SiteN1m SiteN2m SiteL1  SiteL2  …
Good    2       0.3136  0.2136  …  3       3       3       2       69      47      …
Bad     1       0.0533  0.1000  …  1       1       1       1       8       15      …
Average 1.5000  0.1835  0.1568  …  2.0000  2.0000  2.0000  1.5000  38.500  31.000  …
```

For the first calculation, the column `Quality` of the file `test_anno5.txt` was used for grouping the sequences. Therefore, the file was provided twice: as the first annotation (`-a1 test_anno5.txt`) and as the group mapping (`-m test_anno5.txt`); the file `test_anno6.csv` was also provided twice: as the second annotation (`-a2 test_anno6.csv`) and as the sequence length database (`-s test_anno6.csv`). The column `Quality` in the former file has number 3, while the SiDs are in the column `ID` with the number 2, which results in `-mc 2,3`. Providing the flag `-nOg/--non_overlapping_groups` allows for omitting the GIDs in the annotation files (in fact, it would be not possible to specify these GIDs in the second annotation file, as the quality information is missing there). Therefore, only three columns are read from each of the annotations: start position, end position, and SID. In the file `test_anno5.txt`, the start positions are in the column number 5, the end positions are in the column number 6, and the SIDs are in the column number 2, which results in `-a1c 5,6,2`. Following the same logic, the option for the second annotation is `-a2c 6,7,2`. The sequence length values are provided in the same file in the column number 3, which results in `-sc 2,3`.

The next two calculations are conducted in similar way:

```
$python slalom.py -a1 test_anno5.txt -a1h 1 -a1c 5,6,2 -a1d ";" -a2 test_anno6.csv -a2h
2 -a2c 6,7,2 -a2d "," -s test_anno6.csv -sh 2 -sc 2,3 -sd "," -m test_anno6.csv -mh 2 -
mc 2,1 -md "," -o output.tsv -nOg -c --quiet
$cat output.tsv
Group   Nseq    P1      P2      …  SiteN1  SiteN2  SiteN1m SiteN2m SiteL1  SiteL2  …
Natural 1       0.5200  0.1500  …  2       1       2       1       52      15      …
Artif.  2       0.0926  0.1741  …  2       3       2       2       25      47      …
Average 1.5000  0.3063  0.1620  …  2.0000  2.0000  2.0000  1.5000  38.500  31.000  …
$python slalom.py -a1 test_anno5.txt -a1h 1 -a1c 5,6,2 -a1d ";" -a2 test_anno6.csv -a2h
2 -a2c 6,7,2 -a2d "," -s test_anno6.csv -sh 2 -sc 2,3 -sd "," -o output.tsv --
sequences_as_groups -c --quiet
$cat output.tsv
Seq.    Nseq    P1      P2      …  SiteN1  SiteN2  SiteN1m SiteN2m SiteL1  SiteL2  …
ABC     1       0.5200  0.1500  …  2       1       2       1       52      15      …
DEF     1       0.1417  0.2667  …  1       2       1       1       17      32      …
XYZ     1       0.0533  0.1000  …  1       1       1       1       8       15      …
Average 1.0000  0.2383  0.1722  …  1.3333  1.3333  1.3333  1.0000  25.667  20.667  …
```

9. Time series as input

SLALOM is a generic statistical tool. Although it was originally developed to work with SCEs in protein and DNA sequences, its abstract design allows to apply it to virtually any kind of sequences, in biology and beyond.

Working with time series follows the same logic as analyzing the ranges in alphabetical sequences. Indeed, time intervals can be viewed as SCEs in the sequences of time units. For processing time series, SLALOM is started with providing the key `-t/--time_unit`. The four following time units with the corresponding values are supported: second (`sec`), minute (`min`), hour (`hour`), and day (`day`). Now, the start and end positions in the annotation files are replaced with time stamps. For the sequence length database, the length values are replaced with pairs of sequence start and end time stamps (so that the sequence length database now contains three columns instead of two). The following two date formats are supported: `dd.mm.YYYY` and `mm/dd/YYYY`. The optional time is expected after the space in the format `HH:MM` or `HH:MM:SS`; AM/PM classification is currently not supported. If the time is not specified, `00:00:00` is assumed.

After the internal conversion of time intervals to symbols, the workflow and the output are the same as for a general SLALOM case. Time sequences as well as individual CSEs can also overlap and the usual rules for overlap resolving will be applied:

```
$cat test_time_lines.tsv
PlaceA  02/01/2015 22:45        02/02/2015 09:15
PlaceB  02/02/2015 02:10        02/03/2015 00:45
PlaceC  02/03/2015 17:05        02/03/2015 22:10
$cat test_anno_events1.tsv
PlaceA  02/01/2015 23:05        02/01/2015 23:30
PlaceA  02/02/2015 03:10        02/02/2015 04:00
PlaceB  02/02/2015 03:15        02/02/2015 04:05
$cat test_anno_events2.tsv
PlaceA  02/01/2015 23:10        02/01/2015 23:40
PlaceB  02/02/2015 03:15        02/02/2015 04:10
PlaceB  02/02/2015 03:45        02/02/2015 05:00
PlaceC  02/03/2015 18:35        02/03/2015 19:10
$python slalom.py -t min -s test_time_lines.tsv -a1 test_anno_events1.tsv -a2
test_anno_events2.tsv -o output.tsv --sequences_as_groups -sum -c --quiet
$cat output.tsv
Seq.    Nseq    P1      P2      …  SiteN1  SiteN2  SiteN1m SiteN2m SiteL1  SiteL2  …
PlaceA  1       0.1190  0.0476  …  2       1       1       1       75      30      …
PlaceB  1       0.0369  0.0775  …  1       2       1       2       50     105     …
PlaceC  1       0.0000  0.1148  …  0       1       0       0       0       35      …
Average 1.0000  0.0520  0.0800  …  1.0000  1.3333  0.6667  1.0000  41.667  56.667  …
```

```
Sum      3                                  …  3        4        2        3        125      170      …
```

Sequence grouping and all the previously discussed associated option work also for time series.