

NAME

sylamer – fast computation of word biases in ranked DNA sequences

SYNOPSIS

```
sylamer -fasta mouse.utrs -universe mouse.rank -k 7 -grow 100\  
-words mouse.seeds.7 -o out.mouse.7  
sylamer -fasta mouse.utrs -universe mouse.rank -k 7 -grow 100\  
-words mouse.seeds.7 -m 4 --m-anchor -o out.mouse.7
```

The `sylamer` program computes P-values associated with small word occurrences in a ranked sequence repository, enabling the discovery and assessment of enrichment and depletion patterns. It emits both cumulative hypergeometric and binomial P-values. In its main mode of invocation it accepts a sequence file in FASTA format, a file containing ranked identifiers, a word length **k**, a step size **S**, and optionally a file containing a list of words of length **k**, as in the examples above. If no word file is specified `sylamer` will construct the full list of words of length **k**. The maximum word length currently supported is 15.

With these parameters, the program will output cumulative hypergeometric and binomial P-values for each word, for each leading window of the ranked list of identifiers, where the windows are nesting and grow with the step size **S**. The model can optionally correct for composition biases using a Markov model on any smaller word size.

The program produces unfiltered results, that have to be further processed to gauge whether any of these results are significant. The program was designed to be fast and yield comprehensive results. Different scenarios have different filtering requirements and may require different approaches towards correction for multiple testing. The user should be aware of this. An approach common in analysis of microRNA seed match biases in UTRs ranked according to mRNA expression fold change is to plot enrichment and depletion values as log-transformed P-values on the positive and negative Y-axes, respectively [1]. In this scenario the plot represents the full set of unfiltered results in an intuitive way. The program is shipped with an R-script that provides this functionality.

Three important parameters are **-fasta**, **-universe**, and **-subset**. The first is required, and specifies the sequence repository to be read. The latter two are mutually exclusive. In the first case, the option specifies a file containing a comprehensive list of ranked identifiers. Any sequence in the repository for which its identifier is not found in this list is purged from the repository. When **-subset** is used, nothing is removed from the repository. The entire repository, including sequences not present in the subset, serves as the background. In both cases, an identifier for which no sequence is found is purged. In the text below, for each of the two cases, the resulting set of sequences is alternatively called the *sequence universe*, the *background*, or simply the *sequence repository*.

The **-subset** option can be used to analyze a set of sequences that are not ranked, relative to a larger background (specified with the **-fasta** option), simply by making it use a single bin.

The program has many options that control its behaviour. Crucially, it can correct for small words composition biases using a Markov model (**-m** and **--m-anchor**). It has further options to control window granularity, to perform random trials on shuffled sequences, to consider both strands, to change output format, and more.

By default simple 1-repeats and 2-repeats are ignored.

The alphabet accepted by the program is A, C, G, T, U, N, X. The bases T and U are treated as equivalent. In the output only T's are reported. Any base not in the alphabet will lead to a message reporting the number of unrecognized characters in the relevant sequence, unless

the option **--funny-ok** is used. The program transforms lowercase characters to uppercase and in the output always uses uppercase.

Section **DESCRIPTION** contains more information. Refer to [1] for an in-depth description of how *sylamer* works, and please cite [1] if you use it in scientific publications.

DESCRIPTION

The *sylamer* program computes P-values associated with small word occurrences in a sequence repository. In each separate run the word size is fixed, as specified with the **-k** option. By default this value is six. The program computes both cumulative hypergeometric and binomial P-values. Hypergeometric P-values are generally considered to be more accurate and have nice symmetry properties [1].

In many cases, the output is most useful when plotted as lines of log-transformed P-values. The program is shipped with an R-script that provides a convenient wrapper implementing this functionality.

P-values are computed for each word separately by looking at the number of its occurrences in a given window and assessing it relative to the total number of counts in the entire sequence repository (called the background), using a cumulative hypergeometric model.

The cumulative hypergeometric model (which measures probabilities when drawing without replacement) requires four parameters. These are: (a) the universe size (the number of balls in a vase), (b) the number of instances of type I (the number of black balls), (c) the sample size (the number of balls drawn from the vase), and (d) the number of instances in the sample of type I (the number of black balls drawn). Given these parameters we compute the probability of drawing at least the number of balls under (c), and the probability of drawing at most that number. The first is an enrichment P-value, the second a depletion P-value. The smallest of these is reported.

When the program runs, each pair of window and word it considers specifies a new cumulative hypergeometric model, where each new window defines a new sample. In this model the universe size is by default the total number of words of length **k** in the background and does not vary across words or windows. The sample size is the number of words of length **k** in the window. This is the same for all words in a given window. The number of type I instances for a given word **w** in the background is the number of times **w** occurs in the background. This number depends on **w** only. The number of type I instances in the sample for **w** is the number of times that **w** occurs in a given window.

It should be noted that an enrichment hypergeometric P-value in a leading bin of a sequence set for a given word is exactly the hypergeometric depletion P-value for that word for the complement trailing bin, as a result of the properties of drawing without replacement.

The R-script shipped with *sylamer* tracks enrichment scores across all bins computed. In case of enrichment, the absolute log P-value is measured along the positive y-axis. In case of depletion, the log P-value is measured along the negative y-axis. In conjunction with the symmetry property described above, this implies (assuming that bin offsets are symmetrically distributed across the sequence set) that these plots are the mirror image (mirroring in both x- and y-axis, or equivalently rotating the plot 180 degrees) of the plot resulting from using the reversed sequence ranking.

OPTIONS

-k <int> (*word length K*)

The word length to use. By default this has value 6. If either **-words** or **-read-expected** is used, the program will check that the word size found in the file argument agrees with this option. In particular, if the word file contains words of any length other than 6, it is

necessary to specify the same word length using this option.

-fasta <fname> (*fasta file*)

-universe <fname> (*file containing ranked universe IDs*)

-subset <fname> (*file containing ranked subset IDs*)

-o <fname> (*result file*)

These options are the primary input and output options. In order to specify STDIN or STDOUT for any of them, use a single hyphen (-).

The **-fasta** option is required, and should specify a file in FASTA format containing DNA or RNA sequences. Either **-universe** or **-subset** should be given. In both cases, the argument specifies the name of a file containing a single identifier per line. Any identifier not present in the FASTA file is disregarded. Then, with **-subset**, the sequences in the subset thus specified will be analyzed relative to the entire set of sequences in the FASTA file. This primarily means that the background counts are derived from the FASTA file, and that the analysis only considers the subset.

With **-universe** the sequences from the FASTA file will be purged so that they only include identifiers found in the universe file. This should preferably be done when the universe file represents a comprehensive set of sequences. With this analysis, if a fully windowed approach is taken, the last window will coincide with the universe, and all cumulative hypergeometric P-values should equal one (as there is only one possible outcome when the sample size equals the universe size).

The **-o** option specifies the result file. This also applies when one of the **--table** or **--dump-extremes** options is used.

-m <int> (*conditionalize on word length <int>*)

--m-anchor (*modulate by u_r / u_e*)

With **-m** composition biases are corrected using the observed occurrences in the current window of words of the specified, smaller size. From these observed occurrences it will, using a Markov model, derive expected frequencies of occurrence of the words (of length **k**) that it is testing. For a given word, this value is then used to compute the expected number of word occurrences in the background, and the latter replaces the actual occurrence count of that word in the hypergeometric formula.

If the window size approaches the universe size and the expected counts deviate from the real counts, this approach may lead to wildly ballooning P-values. This can be mitigated by multiplying the expected count with a factor u_r / u_e , where u_r is the number of observed counts in the universe and u_e is the number of expected counts in the universe. This behaviour is activated with the **-m-anchor** option, and is recommended whenever **-universe** is used.

-words <fname> (*read words to search from file*)

The words thus read should all have equal length and be valid DNA or RNA words. Each word should occur on a line of its own at the beginning of the line, and each line should specify a word in this manner. The line may contain more information. It will be ignored, as *sylamer* simply takes the first word on the line, delimited by white space. A common application is to specify the list of all microRNA seed matches of a given length for a given species. The **-k** option should be set so that it matches the length of the words in the file.

The output word order is the same as in the word file.

- grow** <int> (*analyze nested bins of size <int>*)
- step** <int> (*analyze consecutive bins of size <int>*)
- step-times** <int> (*analyze the subset in <int> consecutive bins*)
- grow-times** <int> (*analyze the subset in <int> nested bins*)

With **-grow** N the first bin is started by taking the first N sequences from the repository according to the ranking specified (if any). For the next bin the next N sequences are added and so on. The results thus pertain to growing windows, each window being a prefix of its successor window, and the latter extending the former with N sequences.

The option **-grow-times** M is similar to **-grow** in how windows are related, only its argument specifies in advance the *number* of windows to use rather than the window increment size. The appropriate window increment size is computed, and if necessary, this number will be slightly varied at regular intervals so that the sequence universe is exhausted.

With **-step** N windows will be consecutive and non-overlapping. For ranked universes this option is generally not appropriate.

As before, the option **-step-times** M is similar to **-step** in how windows are constructed, only its argument specifies in advance the *number* of windows to use rather than the window size. The program behaviour is then the same as describe under the **-grow-times** option.

If you want the last bin size increments for the trailing bins to be of the exact size as specified on the command line use the option **-aa 2**. This is a special case usage of the option **-aa**, described further below. The result will be that if a bin increment of irregular size is present it will be fit in the middle of the sequence universe, rather than at the end. A bin increment of irregular size will be present unless the universe size is a perfect multiple of the bin increment. Using **-aa** makes the bin offset distribution maximally symmetric with respect to the ulterior ends of the ranked sequence set.

- a** <int> (*accelerate bin increment up until <int>-fold*)
- aa** <int> (*defer acceleration <int>-times*)

These options enable different bin increments across the ranked sequence universe. This can be useful if the universe is large, such as the set of 3' UTRs for a genome. The purpose is to speed up the program. This can be especially useful when generating initial results for many experiments. For publications it is advised not to use these options.

In the leading and trailing portions of the ranked sequence set it is generally advisable to use a small bin increment such as 50 or 100, as occurrence biases tend to fluctuate more in these regions. A finer sampling of these fluctuations may establish a better estimate of the portion of sequences contributing to enrichment or depletion. Near the middle of the ranked sequence set occurrence biases tend to fluctuate less. It is uncommon for a sequence set to divide into two large fractions, exhibiting depletion and enrichment of a particular word, with a well-defined boundary between them.

The **-a** option instructs *sylamer* to accelerate the bin increment in multiples of the initial bin size as specified with **-grow** or **-step**. If we assume an initial bin size S , the second bin increment will be $2S$, the third bin increment will be $3S$, and so forth. With **-grow** and its overlapping windows this will lead to window sizes respectively S , $3S$, and $6S$. As it is often preferable to have a finer sampling in the ulterior parts of the ranking, this acceleration can be deferred a number of times, as specified with the **-aa** option. With the setting `-grow 50 -a 5 -aa 5` the initial bin sizes will be

50 100 150 200 250 350 500 700 950 1200 1450 ..

The opposite behaviour is enforced as the trailing portion of the ranked sequences is reached, and the bin offset distribution is made as symmetric as possible. Thus, with the settings `-grow 2 -a 3 -aa 3` and a sequence universe containing 35 sequences, the full set of bin sizes will be

2 4 6 10 16 19 25 29 31 33 35

-do <int> (*stop after computing bin <int>*)

This option causes the program to compute results for at most <int> bins. It does not affect the way in which bin sizes are computed.

-shift <int> (*ignore the first and last <int> sequences*)

-lshift <int> (*ignore the first <int> sequences*)

-rshift <int> (*ignore the last <int> sequences*)

For these options, *first* and *last* has to be taken in the context of the ranked query file specified with either **-subset** or **-universe**. It is possible to ignore leading and trailing portions of these files using the options above. These options are mostly for research purposes, as the importance of leading and trailing portions can generally be learned from the usual output.

--reverse (*reverse the input order (subset or universe)*)

This reverses the input order. When plotting with the sylamer-associated R-script, this should create a perfect mirror image (reflected in both x and y axis) provided that the bin offsets are symmetrically positioned across the ranked sequence set.

-write-missing <fname> (*write missing IDs to file*)

-write-found <fname> (*write found IDs to file*)

This refers to identifiers from the file specified with either **-universe** or **-subset** which are not found in the sequence repository (given with **-fasta**). Potentially useful for debugging identifier files.

--over (*only do overrepresentation (right/upper tail)*)

--under (*only do underrepresentation (left/lower tail)*)

--none (*skip both over and under-representation*)

--no-binomial (*no binomial*)

These options mainly exist to test various run-time aspects of the program. The **--no-binomial** option can be useful if every ounce of performance has to be squeezed out of *sylamer* and binomial P-values are not of interest.

--two-strands (*consider both strands*)

It is possible to inspect sequences in both strand directions. With this option the reverse complement of a sequence is added to the repository, and the two complementary sequences sort together as a pair. This doubles the sequence repository. All scores for pairs of words that are reverse complements should be identical. Both scores for a pair of complements are reported.

--dump-extremes (*output best P-value over all bins*)

--shuffle (*shuffle universe*)

-trial <int> (*run repeated trials, <int> times*)

The first option causes only the smallest P-value across all bins to be output for a given word. This may be either a depletion or an enrichment value. This option is mainly

intended to be used in conjunction with **-trial**, to gauge the expected levels of enrichment and depletion in random trials. The **--shuffle** option randomly shuffles the universe.

With **-trial**, *sylamer* will randomly shuffle the sequence universe, enable the mode implied by **--dump-extremes**, and repeat this procedure *<int>* times.

-co *<num>* (*P-value threshold*)

Results are output only if the associated P-value is smaller than the cutoff as specified. This will break any subsequent application of plotting.

-dump-expected *<prefix>* (*prefix for dumping expected frequencies*)

-read-expected *<fname>* (*read in expected frequencies*)

The first option will cause the output of expected frequencies based on the chosen Markov order for each bin that is computed. The argument serves as a prefix from which the corresponding file names are constructed. Additionally, a file is created that contains the real observed sequences in *<prefix>.real*.

With the second option *sylamer* reads in a file that contains expected occurrence frequencies. The format has to be the same as the one generated when using **-dump-expected**. It is possible to read in a partial list of word-frequency pairs, similar to the usage of the **-words** option. In that case, only words that are present in the input file will be considered.

-u *<num>* (*fake universe size when computing expected values*)

-u-times *<num>* (*fake universe size as multiple of subset size*)

These options are intended to enable control over the universe size when no representative sequence universe is available. The first option sets the universe size to the specified value. The second option computes it as a multiple of the size of the subset specified with the **-subset** option.

-unit-size *<num>* (*fake unit size / simulate equal sequence lengths*)

This option can be used to test the hypothesis that length normalization is not appropriate. Normally a window size is computed as the number of words found in the sequences contained in the window. Longer sequences generally have more words. The hypothesis that length normalization is not appropriate presumes that solely the presence of a word in a given sequence is what matters, irrespective of the sequence length. Use this option with care.

-threshold *<num>* (*count per-sequence occurrences exceeding <num>*)

The program changes its behaviour quite drastically with this option. It no longer counts word occurrences. Instead, it counts, for each word, the number of times it sees a sequence that has *at least* *<num>* occurrences of the word. The window (sample) size is simply taken as the number of sequences in the window. This option can be used to test the hypothesis that the *number* of matches a given word has in a sequence is of primary concern, irrespective of sequence length. Use this option with care.

-v *<int>* (*set verbosity level*)

By default it is set to 1. The processing of each bin is then signified by the program emitting a dot to STDERR. Set it to 0 to silence *sylamer*. Set it to 2 to obtain more information for each bin that was processed.

-log *<fname>* (*log file*)

Use this to specify a log file. If strange events occur they are written to this file. One such an event is for example when the window occurrence count of a word exceed the

expected background count. This may happen simply because a word occurs much more often than expected given the model underlying the expected frequencies.

--r2-off (*do not disregard 1-shift or 2-shift repeats*)

By default subsequent occurrences of words in 1-repeats or 2-repeats are ignored. Only one of the repeated occurrences is taken into consideration and the program pretends that the attached repeats are masked. An example of a 1-repeat is AAAAAAAAAA, an example of a 2-repeat is GCGCGCGCGCGCG.

--ck (*count sylamer space in units of k*)

Window sizes are by default counted as the total number of words that are available within the contained sequences. A fully unmasked stretch of sequence of length 100 thus contains 94 words of length 7. Any masked base introduces a sequence boundary, as any word that contains a masked base is not considered.

This option changes the way window sizes are counted to the total number of nonoverlapping words of length **k** that fit in the sequences for that window. The unmasked stretch of sequence of length 100 contains 14 words of length 7 in this way of counting.

--funny-ok (*do not warn on seeing strange bases*)

Normally a warning is issued when a sequence contains a base other than A, C, G, T, U, N, or X. Use this option to turn those warnings off.

--table (*collect results per word*)

By default all results are output as they are computed. This means that when plotting the values for a given word are dispersed throughout the output, as each bin generates results for all words. The plotting code thus has to collect the output and restructure it.

This option obviates this need in one special case, namely that of the hypergeometric P-values. It saves all of these in a table structure that is assembled while the program is computing P-values. When finished, the table is output, each row representing the results for a single word over all bins. Each row contains only the results for the hypergeometric P-values. They are log-transformed. For overrepresentation values the absolute log-value is returned, and for underrepresentation values the negative absolute log-value is returned, in accordance with the customary plotting conventions.

NOTE

The first row of the table represents a list of column headers. Each column represents a particular bin, and the header for that column contains the size of that bin (when using nested bins) or the cumulative size of all bins so far (when using stepwise bins). This information should be used to find the location on the x-axis where the values for each bin should be plotted. This is for example relevant when using the **-a** option.

--legend (*write legend for default output format*)

The default output format contains hypergeometric and binomial P-values, the input values for the cumulative hypergeometric distribution, and per-base frequencies of the word under consideration. This option writes the full legend to the result file as the first line, containing all column headers.

-fasta-out <fname> (*fasta output file*)

-rc-out <fname> (*reverse complement fasta output file*)

These options are mainly for testing the various manipulations of the sequence repository.

--clean-up (*release all memory when done*)

This is to test whether *sylamer* is well-behaved in its memory management. When enabled, the program should free all the memory it has allocated before exiting.

--version (*version information, acknowledgements*)**--help** (*print short option synopsis*)**-h** (*print short option synopsis*)

These are customary options. The first prints version, copyright, and author information. The help options print a short synopsis for every option available.

DIAGNOSTICS

The program prints some summary information to STDERR. For interactive use this can be quite useful. This can be turned off, refer to the **-v** option.

AUTHOR

Stijn van Dongen. Cei Abreu-Goodger wrote the R-script that ships with *sylamer*.

ACKNOWLEDGEMENTS

Cei Abreu-Goodger: design and use suggestions, testing, debugging.

Anton Enright: design and use suggestions, suggestion to use C.

Jacques van Helden: author of oligo-analysis, a related program.

Weldon Whitener: OverSite, bootstrap.

Sergei Manakov: suggestions, testing.

Harpreet Saini: all things microRNA.

Russell Grocock: proposed OverSite, base feature set.

The GSL people: high quality numerical computing.

REFERENCES

[1] Stijn van Dongen, Cei Abreu-Goodger and Anton J. Enright. *Fast assessment of microRNA binding and siRNA off-target effects from expression data*, **submitted**.

NOTES

This page was generated from **ZOEM** manual macros, <http://micans.org/zoem>.