# Package: mire (via r-universe)

October 12, 2026

**Title** Microbiome Integrated Reconstruction and Estimation

**Version** 0.0.1.9000

**Description** An integrated framework for microbiome reconstruction from sequencing data. It leverages tools like Kraken2 for taxonomic classification and combines cell barcodes, UMIs, and k-mer-based quantification to reconstruct microbial signals. Designed for both bulk and single-cell sequencing data, the package enables taxonomic and quantitative profiling of microbial communities.

**License** MIT + file LICENSE

**URL** https://github.com/WangLabCSU/mire

**BugReports** https://github.com/WangLabCSU/mire/issues

**Depends** R (>= 4.2)

**Imports** mirai, cli, rlang, ggplot2, utils

**ByteCompile** true

**Encoding** UTF-8

**Config/build/copy-method** link

**Config/roxygen2/version** 8.0.0

**Roxygen** list(markdown = TRUE)

**SystemRequirements** Cargo (Rust's package manager), rustc, kraken2

**Config/pak/sysreqs** libclang-dev

**Repository** https://wanglabcsu.r-universe.dev

**Date/Publication** 2026-10-11 21:58:10 UTC

**RemoteUrl** https://github.com/WangLabCSU/mire

**RemoteRef** HEAD

**RemoteSha** c0eee75eeb43f6059256f574cd612c5cc479e006

# Contents

blsd . . . 2
denoise_counts . . . 3
koutreads . . . 4
kractor_koutput . . . 6
kractor_reads . . . 8
kraken2 . . . 9
krcount . . . 10
read_kreport . . . 11
rpmm_quantile . . . 12
seq_range . . . 13
seq_refine . . . 14
slsd . . . 16
subseq_actions . . . 17

**Index** **18**

---

`blsd` *Barcode level signal denoising*

---

## Description

True taxa are detected on multiple barcodes and with a proprotional number of total and unique k-mer sequences across barcodes, measured as a significant Spearman correlation between the number of total and unique k-mers across barcodes. (`padj < 0.05`)

## Usage

```
blsd(
  kmer,
  method = "spearman",
  ...,
  min_kmer_len = 3L,
  min_cells = 3L,
  p.adjust = "BH"
)
```

## Arguments

| | |
|---|---|
| `kmer` | kmer file returned by `sckmer()`. |
| `method` | A character string indicating which correlation coefficient is to be used for the test. One of "pearson", "kendall", or "spearman", can be abbreviated. |
| `...` | Other arguments passed to cor.test. |
| `min_kmer_len` | An integer, the minimal number of kmer to filter taxa. SAHMI use `2`. |
| `min_cells` | An integer, the minimal number of cell per taxid. SAHMI use `4`. |
| `p.adjust` | Pvalue correction method, a character string. Can be abbreviated. Details see p.adjust. |

## Value

A polars DataFrame

## See Also

https://github.com/sjdlabgroup/SAHMI

---

`denoise_counts` *Denoise and Quantify Taxa from Kraken2 Results*

---

## Description

This function integrates three signal denoising strategies—cell-line-based contaminant filtering, sample-level correlation filtering, and barcode-level correlation testing— to identify reliable microbial taxa and quantify their abundance across samples. It returns a table of taxa counts (per barcode), after filtering out likely contaminants and false positives.

## Usage

```
denoise_counts(
  kreports,
  kmers,
  umis,
  samples = NULL,
  taxon = c("d__Bacteria", "d__Fungi", "d__Viruses"),
  drop_unmatched_taxa = TRUE,
  quantile = 0.95,
  alpha = 0.05,
  min_reads = 3L,
  min_minimizer_n_unique = 3L,
  min_samples = 3L,
  min_kmer_len = 3L,
  min_cells = 3L,
  cor_threshold = 0,
  p_threshold = 0.05,
  padj_threshold = 0.05,
  ...,
  study = "current study",
  alternative = "greater",
  p.adjust = "BH",
  method = "spearman"
)
```

## Arguments

| | |
|---|---|
| kreports | A character vector of paths to Kraken2 report files, typically output by sckmer(), one per sample. |
| kmers | A character vector of paths to k-mer quantification files, typically output by sckmer(), one per sample. |
| umis | A character vector of paths to UMI quantification files, typically output by sckmer(), one per sample. |
| samples | A character of sample identifier for each element in umis. If not provided, the names of the umis vector will be used. |
| drop_unmatched_taxa | A boolean value, indicates whether taxa not found in celllines data should be dropped. Default: TRUE. |
| quantile | Probabilities with values in [0, 1] specifying the quantile to calculate. |
| alpha | Level of significance. |
| min_reads | An integer, the minimal number of the total reads to filter taxa. SAHMI use 2. |
| min_minimizer_n_unique | An integer, the minimal number of the unique number of minimizer to filter taxa. SAHMI use 2. |
| min_samples | An integer, the minimal number of samples per taxid. SAHMI use 4. |
| min_kmer_len | An integer, the minimal number of kmer to filter taxa. SAHMI use 2. |
| min_cells | An integer, the minimal number of cell per taxid. SAHMI use 4. |
| cor_threshold | Minimum correlation coefficient required in sample-level filtering. Default is 0. |
| p_threshold | Significance threshold for sample-level correlation p-values. Default is 0.05. |
| padj_threshold | Adjusted p-value threshold for barcode-level denoising. Default is 0.05. |
| ... | Other arguments passed to cor.test. |
| study | A string of the study name, used to differentiate with cell line data. |
| alternative | A string specifying the alternative hypothesis, must be one of "two.sided", "greater" (default) or "less". You can specify just the initial letter. |
| p.adjust | Pvalue correction method, a character string. Can be abbreviated. Details see p.adjust. |
| method | A character string indicating which correlation coefficient is to be used for the test. One of "pearson", "kendall", or "spearman", can be abbreviated. |

---

koutreads *Kraken2 Output Processing for FASTQ Files*

---

## Description

This function processes Kraken2 output and associated FASTQ files, extracting relevant taxonomic information and preparing them for further downstream analysis.

## Usage

```
koutreads(
  kreport,
  koutput,
  reads,
  ofile,
  tag_ranges1 = NULL,
  tag_ranges2 = NULL,
  taxonomy = c("D__Bacteria", "D__Fungi", "D__Viruses"),
  exclude = c("9606"),
  koutput_batch = NULL,
  fastq_batch = NULL,
  chunk_bytes = NULL,
  compression_level = 4L,
  nqueue = NULL,
  threads = NULL,
  odir = NULL
)

tag(tag, ranges)
```

## Arguments

| | |
|---|---|
| `kreport` | Path to the Kraken2 report file. |
| `koutput` | Path to the Kraken2 output file. |
| `reads` | A character vector of FASTQ file paths, either the original reads used as input to Kraken2 or the classified output reads (recommended for efficiency as they are smaller). Accepts one file for single-end or two files for paired-end.<br>**If only one file is used in Kraken2 to generate the koutput file, only the second read sequence will be extracted to match the koutput's Lowest Common Ancestor (LCA)**. This is commonly observed in data such as 10x Genomics, where the first read contains only UMI and barcode information, while the second read carries the biological sequence. You can always change the input files to treat the actual biological read as read1 if needed. |
| `ofile` | A character string. Path to the output file that will store the matched reads extracted based on Kraken2 classification. The output is compressed if the extension is `.gz`. This file contains only reads whose taxonomic assignments match the filtering criteria, such as `taxonomy` inclusion and `exclude` filters. Useful for downstream analysis like quantification of taxon-specific reads. |
| `tag_ranges1, tag_ranges2` | A list of sequence ranges for extracting tags from the first/second read in single-end/paired-end data. If `NULL`, no additional tag extraction occurs for the first/second read. These ranges must be created by the `tag()` function, or be a named `seq_range()` object (in which case it will be automatically wrapped by `tag()`). Note that the tag embedded by `seq_refine()` in the description header will always be extracted. |

taxonomy — A character vector. The set of taxonomic groups to include (default: c("D__Bacteria", "D__Fungi", "D__Viruses")). This defines the global taxa to consider. Only the descendants within these groups will be considered. If NULL, all taxa will be used.

exclude — A character vector of taxids to exclude sequences from usage. Typically used to exclude the host taxid (e.g., 9606 for human) from the analysis. By default, this excludes human sequences ("9606").

koutput_batch, fastq_batch — Integer. Number of FASTQ records/Koutput lines to accumulate before dispatching a chunk to worker threads for processing. This controls the granularity of parallel work and affects memory usage and performance. Default is 1000 for koutput_batch and 256 for fastq_batch.

chunk_bytes — Integer specifying the size in bytes used for compressing and writing records in batches to disk. Default is 8 * 1024 * 1024 (8MB).

compression_level — Integer from 1 to 12 (default: 4). This sets the gzip compression level when writing output files. A higher value increases compression ratio but may slow down writing. Only applies when output filenames end with .gz.

nqueue — Integer. Maximum number of buffers per thread, controlling the amount of in-flight data awaiting writing. Default: 3. Setting this too high may increase memory consumption without performance gain.

threads — Integer. Number of threads to use. Default: 3.

odir — A string of directory to save the output files. Please see Value section for details.

tag — An character label used to label the extracted content.

ranges — A range or a list of ranges specifying the subsequence(s) to process. Must be created using the seq_range() function.

---

kractor_koutput *Filter Kraken2 Output by Taxon*

---

## Description

This function filters Kraken2 classification output (koutput) by taxonomic identifiers derived from a Kraken2 report (kreport). It extracts lines matching the desired taxonomy, ranks, taxa, taxids, and descendants and writes the filtered results to an output file.

## Usage

```
kractor_koutput(
  kreport,
  koutput,
  ofile,
  taxonomy = c("D__Bacteria", "D__Fungi", "D__Viruses"),
  ranks = NULL,
```

```
  taxa = NULL,
  taxids = NULL,
  exclude = NULL,
  descendants = TRUE,
  batch_size = NULL,
  chunk_bytes = NULL,
  compression_level = 4L,
  nqueue = NULL,
  threads = NULL,
  odir = NULL
)
```

## Arguments

| | |
|---|---|
| kreport | Path to the Kraken2 report file. |
| koutput | Path to the Kraken2 output file. |
| ofile | A character string. Path to the output file storing the filtered Kraken2 output lines that pass taxonomic and exclusion filters. If the filename ends with .gz, output will be automatically compressed using gzip. |
| taxonomy | Character vector. The set of taxonomic groups to include (default: c("D__Bacteria", "D__Fungi", "D__Viruses")). This defines the global taxa to consider. If NULL, all taxa will be used. If descendants = TRUE, only the descendants within these groups will be considered. The selection of taxa can be further refined using the ranks, taxa, and taxids parameters. One of taxonomy, ranks, taxa, or taxids must be provided. |
| ranks | Character vector. The taxonomic ranks to filter by (optional). |
| taxa | Character vector. Specific taxa to include (optional). |
| taxids | Character vector. A list of taxid values to filter by (optional). |
| exclude | A character vector of taxids to exclude sequences from usage. |
| descendants | Logical. Whether to include descendants of the selected taxa (default: TRUE). |
| chunk_bytes | Integer specifying the size in bytes used for compressing and writing records in batches to disk. Default is 8 * 1024 * 1024 (8MB). |
| compression_level | Integer from 1 to 12 (default: 4). This sets the gzip compression level when writing output files. A higher value increases compression ratio but may slow down writing. Only applies when output filenames end with .gz. |
| nqueue | Integer. Maximum number of buffers per thread, controlling the amount of in-flight data awaiting writing. Default: 3. Setting this too high may increase memory consumption without performance gain. |
| threads | Integer. Number of threads to use. Default: 3. |
| odir | A string of directory to save the output files. Please see Value section for details. |

## Value

None. The function generates a filtered Kraken2 output file containing entries corresponding to the specified taxonomy, ranks, taxa, taxids, and descendants extracted from the input koutput.

`kractor_reads` *Extract Reads from Kraken2 Output Based on Classification*

## Description

This function extracts reads corresponding to selected classifications from a Kraken2 output file (`koutput`). Only reads classified to selected taxa will be extracted from the provided sequence file (`reads`).

## Usage

```
kractor_reads(
  koutput,
  reads,
  ofile1 = NULL,
  ofile2 = NULL,
  batch_size = NULL,
  chunk_bytes = NULL,
  compression_level = 4L,
  nqueue = NULL,
  threads = NULL,
  odir = NULL
)
```

## Arguments

| | |
|---|---|
| `koutput` | Path to the Kraken2 output file. |
| `reads` | A character vector of FASTQ file paths. Accepts one file for single-end or two files for paired-end. |
| `ofile1` | Output FASTQ file path for the first read (`fq1`). Required when only one input file is given (i.e., single-end mode). Optional when two input files are used. |
| `ofile2` | Optional path to the output FASTQ file for `fq2`. |
| `batch_size` | Integer. Number of FASTQ records to accumulate before dispatching a chunk to worker threads for processing. This controls the granularity of parallel work and affects memory usage and performance. Default is `256`. |
| `chunk_bytes` | Integer specifying the size in bytes used for compressing and writing records in batches to disk. Default is `8 * 1024 * 1024` (8MB). |
| `compression_level` | Integer from 1 to 12 (default: `4`). This sets the gzip compression level when writing output files. A higher value increases compression ratio but may slow down writing. Only applies when output filenames end with `.gz`. |
| `nqueue` | Integer. Maximum number of buffers per thread, controlling the amount of in-flight data awaiting writing. Default: `3`. Setting this too high may increase memory consumption without performance gain. |
| `threads` | Integer. Number of threads to use. Default: `3`. |
| `odir` | A string of directory to save the output files. Please see `Value` section for details. |

kraken2 *Run Kraken2 for Taxonomic Classification*

## Description

This function runs the Kraken2 classifier on one or two FASTQ files (single- or paired-end). It wraps around the `blit::kraken2()` command interface and adds optional support for environment setup (e.g., Conda or custom `PATH`).

## Usage

```
kraken2(
  reads,
  ...,
  db = NULL,
  kreport = "kraken_report.txt",
  koutput = "kraken_output.txt",
  classified_out = "classified.fq",
  unclassified_out = NULL,
  kraken2 = NULL,
  envpath = NULL,
  conda = NULL,
  condaroot = NULL,
  odir = NULL
)
```

## Arguments

| | |
|---|---|
| `reads` | A character vector of FASTQ files used as input to Kraken2. Can be one file (single-end) or two files (paired-end). |
| `...` | <dynamic dots> Additional arguments passed to `kraken2` command. Empty arguments are automatically trimmed. If a single argument, such as a file path, contains spaces, it must be quoted, for example using `shQuote()`. Details see: `cmd_help(kraken2())`. |
| `db` | Path to the Kraken2 database. You can download prebuilt databases from https://benlangmead.github.io/aws-indexes/k2, or build your own by following the instructions at https://github.com/DerrickWood/kraken2/wiki/Manual#kraken-2-databases. |
| `classified_out` | A string of path to save classified sequences, which should be a fastq file. |
| `unclassified_out` | A string of path to save unclassified sequences, which should be a fastq file. |
| `kraken2` | Optional. Path to the Kraken2 binary if not in the system `PATH`. |
| `envpath` | Optional. Additional directory to prepend to the system `PATH` environment variable. Useful when `kraken2` is installed in a non-default location. |
| `conda` | Optional. Name of the Conda environment to activate before running Kraken2. |

condaroot — A string specifying the path to the conda root prefix. If not provided, the function searches for the root in the following order:

1. the option `blit.conda.root`.
2. the environment variable `BLIT_CONDA_ROOT`.
3. the root prefix of `appmamba()`.

odir — A string of path to the output directory.

## Value

None. This function is called for its side effects. It produces the following output files in `odir` (or the working directory if `odir` is `NULL`):

- `kreport`: Kraken2 classification report
- `koutput`: Kraken2 raw classification output
- `classified_out`: FASTQ file of classified reads
- `unclassified_out`: FASTQ file of unclassified reads (if specified)

---

# krcount *Quantify K-mers and UMI Data by Taxon*

---

## Description

This function counts total and unique k-mers per taxon across cell barcodes, using both the cell barcode and unique molecular identifier (UMI) to resolve read identity at the single-cell level. It aggregates k-mer counts for each taxonomic rank of interest (by default, genus and species), including all descendant taxa within those ranks.

## Usage

```
krcount(
  koutreads,
  kreport,
  umi_tag = NULL,
  barcode_tag = NULL,
  taxonomy = c("D__Bacteria", "D__Fungi", "D__Viruses"),
  batch_size = NULL,
  nqueue = NULL
)
```

## Arguments

koutreads — Path to the output file produced by `koutreads()`.

kreport — Path to the Kraken2 report file.

umi_tag — (Optional) A string specifying the tag used to extract unique molecular identifiers (UMIs) from each read. If `NULL`, all reads are counted as total fragments. Otherwise, only unique UMIs per (barcode, taxon) are counted.

barcode_tag (Optional) A string specifying the tag used to extract the cell barcode from each read. If NULL, all reads are assumed to originate from a single cell.

taxonomy A character vector. The set of taxonomic groups to include (default: c("D__Bacteria", "D__Fungi", "D__Viruses")). This defines the global taxa to consider. Only the descendants within these groups will be considered. If NULL, all taxa will be used.

nqueue Integer. Maximum number of buffers per thread, controlling the amount of in-flight data awaiting writing. Default: 3. Setting this too high may increase memory consumption without performance gain.

---

read_kreport *Parse kraken report file*

---

## Description

Parse kraken report file

## Usage

```
read_kreport(kreport, taxonomy = NULL)
```

## Arguments

kreport A string giving the path to a Kraken2 report file. Both the standard six-column format and the eight-column format produced with --report-minimizer-data are supported.

taxonomy An optional character vector of taxonomic groups to include, written as a taxid or "rank__name", for example "562" or "D__Bacteria". Matching taxa and all their descendants are retained. If NULL, every report row, including the unclassified row, is returned.

## Value

A data frame containing the report columns percents, total_reads, reads, rank, taxid, and taxon. Eight-column reports additionally contain minimizer_len and minimizer_n_unique. The list columns ranks, taxids, and taxa contain each row's classified lineage; root and unclassified rows are not included as ancestors.

## See Also

https://github.com/DerrickWood/kraken2/blob/master/docs/MANUAL.markdown

`rpmm_quantile` *Identifying contaminants and false positives taxa (cell line quantile test)*

## Description

This function detects likely contaminant or false-positive microbial taxa by comparing read abundance (in reads per million microbiome reads, RPMM) from input Kraken2 reports with a curated reference of known contaminants derived from cell lines. It performs a one-sample quantile test on each taxon and reports taxa with significant enrichment above reference levels.

## Usage

```
rpmm_quantile(
  kreports,
  study = "current study",
  taxon = c("d__Bacteria", "d__Fungi", "d__Viruses"),
  quantile = 0.95,
  alpha = 0.05,
  alternative = "greater",
  drop_unmatched_taxa = TRUE
)
```

## Arguments

| | |
|---|---|
| `kreports` | A character of path to all kraken report files. |
| `study` | A string of the study name, used to differentiate with cell line data. |
| `quantile` | Probabilities with values in `[0, 1]` specifying the quantile to calculate. |
| `alpha` | Level of significance. |
| `alternative` | A string specifying the alternative hypothesis, must be one of "two.sided", "greater" (default) or "less". You can specify just the initial letter. |
| `drop_unmatched_taxa` | A boolean value, indicates whether taxa not found in celllines data should be dropped. Default: `TRUE`. |

## Value

A polars DataFrame with following attributes:

1. `pvalues`: Quantile test pvalue.
2. `unmatched_taxa`: taxids in current study but not found in cellline data.
3. `significant`: significant taxids with `pvalues` < `alpha`.
4. `truly`: truly taxids based on `alpha` and `unmatched_taxa`. If `drop_unmatched_taxa` isn't `TRUE`, this should be the union of `unmatched_taxa` and `significant`, otherwise, this should be the same with `significant`.

## Examples

```
## Not run:
# `paths` should be the output directory for each sample from
# `blit::kraken2()`
quantile_test <- remove_contaminants(
    kreports = file.path(paths, "kraken_report.txt"),
    quantile = 0.99, drop_unmatched_taxa = TRUE
)
ggplot2::autoplot(quantile_test)

## End(Not run)
```

---

`seq_range` *Create a UMI/Barcode range*

---

## Description

Constructs a `mire_seq_range` object representing a UMI/Barcode range with optional start and end positions.

## Usage

```
seq_range(start = NULL, end = NULL)
```

## Arguments

| | |
|---|---|
| `start` | Integer. Start position (1-based). Optional if `end` is provided. |
| `end` | Integer. End position (1-based). Optional if `start` is provided. |

## Value

A `mire_seq_range` or `mire_seq_ranges` object.

## Examples

```
# Create a single range
seq_range(start = 100, end = 200)

# If `end` is NULL, the range goes to the end of the sequence
seq_range(start = 100)

# If `start` is NULL, the range starts from the beginning of the sequence
seq_range(end = 200)

# Combine multiple ranges
r1 <- seq_range(100, 200)
r2 <- seq_range(300, 400)
r3 <- seq_range(end = 500)
```

```
# Combine them into a multi-range object using `c()`
c(r1, r2, r3)

# Nested combinations are also supported
c(c(r1, r2), r3)
```

---

`seq_refine` *Refine Sequences from FASTQ Files with UMI/Barcode Actions*

---

## Description

This function refines one or two FASTQ files by applying trimming and embedding operations defined via user-specified actions such as `embed()`, `trim()`, or `embed_trim()`. It supports single-end and paired-end reads, processes data in chunks, and uses multithreading for performance.

## Usage

```
seq_refine(
  reads,
  ofile1 = NULL,
  ofile2 = NULL,
  umi_action1 = NULL,
  umi_action2 = NULL,
  barcode_action1 = NULL,
  barcode_action2 = NULL,
  extra_actions1 = NULL,
  extra_actions2 = NULL,
  batch_size = NULL,
  chunk_bytes = NULL,
  compression_level = 4L,
  nqueue = NULL,
  threads = NULL,
  odir = NULL
)
```

## Arguments

| | |
|---|---|
| `reads` | A character vector of FASTQ file paths. Accepts one file for single-end or two files for paired-end. |
| `ofile1` | Output FASTQ file path for the first read (`fq1`). Required when only one input file is given (i.e., single-end mode). Optional when two input files are used. |
| `ofile2` | Optional path to the output FASTQ file for `fq2`. |
| `umi_action1, umi_action2` | Sequence action for extracting or trimming UMI from `fq1`/`fq2`. `umi_action2` is only allowed if `fq2` is provided. Specify sequence ranges using `seq_range()` or combine multiple ranges with `c()`, this also applies to following all actions. By default, the UMI is embedded in the header and trimmed from the sequence |

and quality (embed_trim()). You can customize the behavior with embed(), or trim().

barcode_action1, barcode_action2
: Sequence action for extracting or trimming barcodes from fq1/fq2. barcode_action2 is only allowed if fq2 is provided. By default, barcodes are embedded in the header and trimmed from both the sequence and quality strings (embed_trim()). Specify ranges as with UMI actions, and customize using embed(), or trim().

extra_actions1, extra_actions2
: Additional sequence actions to apply to fq1/fq2. extra_actions2 is only allowed if fq2 is provided. These can be a single one or a list of them. By default, these actions perform trimming of sequences and qualities unless otherwise specified.

batch_size
: Integer. Number of FASTQ records to accumulate before dispatching a chunk to worker threads for processing. This controls the granularity of parallel work and affects memory usage and performance. Default is 256.

chunk_bytes
: Integer specifying the size in bytes used for compressing and writing records in batches to disk. Default is 8 * 1024 * 1024 (8MB).

compression_level
: Integer from 1 to 12 (default: 4). This sets the gzip compression level when writing output files. A higher value increases compression ratio but may slow down writing. Only applies when output filenames end with .gz.

nqueue
: Integer. Maximum number of buffers per thread, controlling the amount of in-flight data awaiting writing. Default: 3. Setting this too high may increase memory consumption without performance gain.

threads
: Integer. Number of threads to use. Default: 3.

odir
: A string of directory to save the output files. Please see Value section for details.

## Details

Actions define what to do with sequence ranges specified using seq_range().

- UMI/barcode actions must be a single range (no lists) and default to embed_trim() if not specified.
- Extra actions can be multiple ranges (use a list) and default to trim() if not specified.
- Use embed(), trim(), or embed_trim() to specify the behavior.

## Value

None. Outputs processed FASTQ files as specified by ofile1 and ofile2.

`slsd` *Sample level signal denoising*

## Description

In the low-microbiome biomass setting, real microbes also exhibit a proportional number of total k-mers, number of unique k-mers, as well as number of total assigned sequencing reads across samples; i.e. the following three Spearman correlations are significant when tested using sample-level data provided in Kraken reports: `cor(minimizer_len, minimizer_n_unique)`, `cor(minimizer_len, total_reads)` and `cor(total_reads, minimizer_n_unique)`. (`r1>0 & r2>0 & r3>0 & p1<0.05 & p2<0.05 & p3<0.05`).

## Usage

```
slsd(
  kreports,
  method = "spearman",
  ...,
  min_reads = 3L,
  min_minimizer_n_unique = 3L,
  min_samples = 3L
)
```

## Arguments

| | |
|---|---|
| `kreports` | kreport files returned by `sckmer()` for all samples. |
| `method` | A character string indicating which correlation coefficient is to be used for the test. One of "pearson", "kendall", or "spearman", can be abbreviated. |
| `...` | Other arguments passed to cor.test. |
| `min_reads` | An integer, the minimal number of the total reads to filter taxa. SAHMI use 2. |
| `min_minimizer_n_unique` | An integer, the minimal number of the unique number of minimizer to filter taxa. SAHMI use 2. |
| `min_samples` | An integer, the minimal number of samples per taxid. SAHMI use 4. |

## Value

A polars DataFrame of correlation coefficient and pvalue for `cor(minimizer_len, minimizer_n_unique)` (r1 and p1), `cor(minimizer_len, total_reads)` (r2 and p2) and `cor(total_reads, minimizer_n_unique)` (r3 and p3).

subseq_actions *Define sequence range behavior: embed, trim, or both*

## Description

These functions annotate a range or a list of ranges to indicate how the specified subsequences (e.g. UMI or barcode) should be handled during FASTQ processing.

## Usage

```
embed(tag, ranges)

trim(ranges)

embed_trim(tag, ranges)
```

## Arguments

| | |
|---|---|
| tag | An character label used when embedding sequence content into the FASTQ header (used with `embed()` and `embed_trim()`).<br>For UMI and barcode actions, the tag will be assigned automatically as `"UMI"` and `"BARCODE"` respectively.<br>For other types of actions, you must explicitly specify a `tag` to ensure clarity in the embedded header. |
| ranges | A range or a list of ranges specifying the subsequence(s) to process. Must be created using the `seq_range()` function. |

## Details

- `embed()` extracts the sequence and appends it to the read header without modifying the sequence.
- `trim()` removes the specified subsequence from the read but does not store it.
- `embed_trim()` both extracts the subsequence for the header and removes it from the read.

Each function wraps the input range object in a new class to indicate its behavior downstream.

## Value

An annotated `mire_seq_range` or `mire_seq_ranges` object. object with behavior-specific class:

- `mire_embed` — for embedding only
- `mire_trim` — for trimming only
- `mire_embed_trim` — for embedding and trimming

# Index


appmamba(), *10*

blsd, 2

cor.test, *2*, *4*, *16*

DataFrame, *3*, *12*, *16*
denoise_counts, 3
dynamic dots, *9*

embed *(*subseq_actions*)*, 17
embed(), *14, 15*
embed_trim *(*subseq_actions*)*, 17
embed_trim(), *14, 15*
environment variable, *10*

koutreads, 4
koutreads(), *10*
kractor_koutput, 6
kractor_reads, 8
kraken2, 9
krcount, 10

option, *10*

p.adjust, *2*, *4*

read_kreport, 11
rpmm_quantile, 12

sckmer(), *2*, *4*, *16*
seq_range, 13
seq_range(), *5, 6*, *14, 15*, *17*
seq_refine, 14
seq_refine(), *5*
shQuote(), *9*
slsd, 16
subseq_actions, 17

tag *(*koutreads*)*, 4
trim *(*subseq_actions*)*, 17
trim(), *14, 15*