Package: rsahmi (via r-universe)

March 27, 2025

Title Single-Cell Analysis of Host-Microbiome Interactions

Version 0.0.2.9000

Description A computational resource designed to accurately detect microbial nucleic acids while filtering out contaminants and false-positive taxonomic assignments from standard transcriptomic sequencing of mammalian tissues. For more details, see Ghaddar (2023) <doi:10.1038/s43588-023-00507-1>. This implementation leverages the 'polars' package for fast and systematic microbial signal recovery and denoising from host tissue genomic sequencing.

License MIT + file LICENSE

ByteCompile true

Encoding UTF-8

OS_type unix

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

SystemRequirements kraken2, seqkit

Imports blit (>= 0.1.0), cli, rlang, ShortRead, utils

Suggests polars (>= 0.17.0)

Additional_repositories https://community.r-multiverse.org

URL https://github.com/Yunuuuu/rsahmi

BugReports https://github.com/Yunuuuu/rsahmi/issues

Config/pak/sysreqs libjpeg-dev libpng-dev libssl-dev

Repository https://yunuuuu.r-universe.dev

RemoteUrl https://github.com/Yunuuuu/rsahmi

RemoteRef HEAD

RemoteSha 85eeeae0e0f61cb2cdc0c17817563451bd994e84

Contents

`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",
  ...,
  p.adjust = "BH",
  min_kmer_len = 3L,
  min_number = 3L
)
```

Arguments

`kmer`	kmer data returned by `prep_dataset()`.
`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.
`p.adjust`	Pvalue correction method, a character string. Can be abbreviated. Details see p.adjust.
`min_kmer_len`	An integer, the minimal number of kmer to filter taxa. SAHMI use 2.
`min_number`	An integer, the minimal number of cell per taxid. SAHMI use 4.

Value

A polars DataFrame

See Also

https://github.com/sjdlabgroup/SAHMI

Examples

```
## Not run:
# 1. `sahmi_datasets` should be the output of all samples from
      `prep_dataset()`
# 2. `real_taxids_slsd` should be the output of `slsd()`
umi_list <- lapply(sahmi_datasets, function(dataset) {
    # Barcode level signal denoising (barcode k-mer correlation test)
    blsd <- blsd(dataset$kmer)
    real_taxids <- blsd$filter(pl$col("padj")$lt(0.05))$get_column("taxid")
    # only keep taxids pass Sample level signal denoising
    real_taxids <- real_taxids$filter(real_taxids$is_in(real_taxids_slsd))
    # remove contaminants
    real_taxids <- real_taxids$filter(
        real_taxids$is_in(attr(truly_microbe, "truly"))
    )
    # filter UMI data
    dataset$umi$filter(pl$col("taxid")$is_in(real_taxids))
})

## End(Not run)
```

`extractor` *Extract reads and output from Kraken*

Description

Extract reads and output from Kraken

Usage

```
extract_taxids(
  kraken_report,
  taxon = c("d__Bacteria", "d__Fungi", "d__Viruses")
)

extract_kraken_output(
  kraken_out,
  taxids,
  odir,
  ofile = "kraken_microbiome_output.txt",
  ...
)

extract_kraken_reads(
```

```
  kraken_out,
  reads,
  ofile = NULL,
  odir = getwd(),
  threads = NULL,
  ...,
  envpath = NULL,
  seqkit = NULL
)
```

Arguments

kraken_report	The path to kraken report file.
taxon	An atomic character specify the taxa name wanted. Should follow the kraken style, connected by rank codes, two underscores, and the scientific name of the taxon (e.g., "d__Viruses")
kraken_out	The path to kraken output file.
taxids	A character specify NCBI taxonony identifier to extract.
odir	A string of directory to save the ofile.
ofile	A string of file save the kraken output of specified taxids.
...	• extract_kraken_output: Additional arguments passed to sink_csv(). • extract_kraken_reads: Additional arguments passed to cmd_run() method.
reads	The original fastq files (input in kraken2). You can pass two paired-end files directly.
threads	Number of threads to use, see blit::cmd_help(blit::seqkit("grep")).
envpath	A string of path to be added to the environment variable PATH.
seqkit	A string of path to seqkit command.

Value

- extract_taxids: An atomic character vector of taxon identifiers.
- extract_kraken_output: A polars DataFrame.
- extract_kraken_reads: Exit status invisiblely.

See Also

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

Examples

```
## Not run:
# For 10x Genomic data, `fq1` only contain barcode and umi, but the official
# didn't give any information for this. In this way, I prefer using
# `umi-tools` to transform the `umi` into fq2 and then run `rsahmi` with
# only fq2.
```

```
blit::kraken2(
    fq1 = fq1,
    fq2 = fq2,
    classified_out = "classified.fq",
    # Number of threads to use
    blit::arg("--threads", 10L, format = "%d"),
    # the kraken database
    blit::arg("--db", kraken_db),
    "--use-names", "--report-minimizer-data",
) |> blit::cmd_run()

# `kraken_report` should be the output of `blit::kraken2()`
taxids <- extract_taxids(kraken_report = "kraken_report.txt")

# 1. `kraken_out` should be the output of `blit::kraken2()`
# 2. `taxids` should be the output of `extract_taxids()`
# 3. `odir`: the output directory
extract_kraken_output(
    kraken_out = "kraken_output.txt",
    taxids = taxids,
    odir = # specify the output directory
)

# 1. `kraken_out` should be the output of `extract_kraken_output()`
# 2. `fq1` and `fq2` should be the same with `blit::kraken2()`
extract_kraken_reads(
    kraken_out = "kraken_microbiome_output.txt",
    reads = c(fq1, fq2),
    threads = 10L, # Number of threads to use
    # try to change `seqkit` argument into your seqkit path. If `NULL`, the
    # internal will detect it in your `PATH` environment variable
    seqkit = NULL
)

## End(Not run)
```

`parse_kraken_report` *Parse kraken report file*

Description

Parse kraken report file

Usage

```
parse_kraken_report(kraken_report, intermediate_ranks = TRUE, mpa = FALSE)
```

Arguments

kraken_report	The path to kraken report file.
intermediate_ranks	A bool indicates whether to include non-traditional taxonomic ranks in output.
mpa	A bool indicates whether to use mpa-style.

Value

A polars DataFrame.

See Also

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

---

prep_dataset *Prepare kraken report, k-mer statistics, UMI data*

Description

Three elements returned by this function:

- kreport: Used by slsd().
- kmer: Used by blsd(). The function count the number of k-mers and unique k-mers assigned to a taxon across barcodes. The cell barcode and unique molecular identifier (UMI) are used to identify unique barcodes and reads. Data is reported for taxa of pre-specified ranks (default genus + species) taking into account all subsequently higher resolution ranks. The output is a table of barcodes, taxonomic IDs, number of k-mers, and number of unique k-mers.
- umi: Used by taxa_counts().

Usage

```
prep_dataset(
  fa1,
  kraken_report,
  kraken_out,
  fa2 = NULL,
  cb_and_umi = function(sequence_id, read1, read2) {
     list(substring(read1, 1L, 16L),
    substring(read1, 17L, 28L))
 },
  ranks = c("G", "S"),
  kmer_len = 35L,
  min_frac = 0.5,
  exclude = "9606",
  threads = 10L,
  overwrite = TRUE,
```

```
  odir = NULL
)

read_dataset(dir)
```

Arguments

fa1, fa2	The path to microbiome fasta 1 and 2 file (returned by extract_kraken_reads()).
kraken_report	The path to kraken report file.
kraken_out	The path of microbiome output file. Usually should be filtered with extract_kraken_output().
cb_and_umi	A function takes sequence id, read1, read2 and return a list of 2 corresponding to cell barcode and UMI respectively., each should have the same length of the input.
ranks	Taxa ranks to analyze.
kmer_len	Kraken kmer length. Default: 35L, which is the default kmer size of kraken2.
min_frac	Minimum fraction of kmers directly assigned to taxid to use read. Reads with <=min_frac of the k-mers map inside the taxon's lineage are also discarded.
exclude	A character of taxid to exclude, for SAHMI, the host taxid. Reads with any k-mers mapped to the exclude are discarded.
threads	Number of threads to use.
overwrite	A bool indicates whether to overwrite the files in odir.
odir	A string of directory to save the results.
dir	A string of directory containing the files returned by prep_dataset.

Value

A list of three polars DataFrame:

- kreport: Used by slsd().
- kmer: Used by blsd().
- umi: Used by taxa_counts().

See Also

https://github.com/sjdlabgroup/SAHMI

Examples

```
# for sequence from `umi-tools`, we can use following function
cb_and_umi <- function(sequence_id, read1, read2) {
    out <- lapply(
        strsplit(sequence_id, "_", fixed = TRUE),
        `[`, 2:3
    )
    lapply(1:2, function(i) {
        vapply(out, function(o) as.character(.subset2(o, i)), character(1L))
```

```
    })
}

## Not run:
# 1. `fa1` and `fa2` should be the output of `extract_kraken_reads()`
# 2. `kraken_report` should be the output of `blit::kraken2()`
# 3. `kraken_out` should be the output of `extract_kraken_output()`
# 4. `dir`: you may want to specify the output directory since this process
#           is time-consuming
sahmi_dataset <- prep_dataset(
    fa1 = "kraken_microbiome_reads.fa",
    # if you have paired sequence, please also specify `fa2`,
    # !!! Also pay attention to the file name of `fa1` (add suffix `_1`)
    # if you use paired reads.
    # fa2 = "kraken_microbiome_reads_2.fa",
    kraken_report = "kraken_report.txt",
    kraken_out = "kraken_microbiome_output.txt",
    odir = NULL
)
# you may want to prepare all datasets for subsequent workflows.
# `paths` should be the output directory for each sample from
# `blit::kraken2()`, `extract_kraken_output()` and `extract_kraken_reads()`.
sahmi_datasets <- lapply(paths, function(dir) {
    prep_dataset(
        fa1 = file.path(dir, "kraken_microbiome_reads.fa"),
        # fa2 = file.path(dir, "kraken_microbiome_reads_2.fa"),
        kraken_report = file.path(dir, "kraken_report.txt"),
        kraken_out = file.path(dir, "kraken_microbiome_output.txt"),
        odir = dir
    )
})

## End(Not run)
```

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

Description

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

Usage

```
remove_contaminants(
  kraken_reports,
  study = "current study",
  taxon = c("d__Bacteria", "d__Fungi", "d__Viruses"),
  quantile = 0.95,
  alpha = 0.05,
```

```
  alternative = "greater",
  exclusive = FALSE
)
```

Arguments

kraken_reports	A character of path to all kraken report files.
study	A string of the study name, used to differentiate with cell line data.
taxon	An atomic character specify the taxa name wanted. Should follow the kraken style, connected by rank codes, two underscores, and the scientific name of the taxon (e.g., "d__Viruses")
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.
exclusive	A boolean value, indicates whether taxa not found in celllines data should be regarded as truly. Default: `FALSE`.

Value

A polars DataFrame with following attributes:

1. `pvalues`: Quantile test pvalue.
2. `exclusive`: 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 `exclusive`. If `exclusive` is `TRUE`, this should be the union of `exclusive` 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()`
truly_microbe <- remove_contaminants(
    kraken_reports = file.path(paths, "kraken_report.txt"),
    quantile = 0.99, exclusive = FALSE
)
microbe_for_plot <- attr(truly_microbe, "truly")[
    order(attr(truly_microbe, "pvalue")[attr(truly_microbe, "truly")])
]
microbe_for_plot <- microbe_for_plot[
    !microbe_for_plot %in% attr(truly_microbe, "exclusive")
]
ggplot(
    truly_microbe$filter(pl$col("taxid")$is_in(microbe_for_plot))$
        to_data_frame(),
    aes(rpmm),
) +
```

```
    geom_density(aes(fill = study), alpha = 0.5) +
    scale_x_log10() +
    facet_wrap(facets = vars(taxa), scales = "free") +
    theme(
        strip.clip = "off",
        axis.text = element_blank(),
        axis.ticks = element_blank(),
        legend.position = "inside",
        legend.position.inside = c(1, 0),
        legend.justification.inside = c(1, 0)
    )

## End(Not run)
```

`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_number = 3L
)
```

Arguments

`kreports`	kreports data returned by `prep_dataset()` 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_number`	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).

Examples

```
## Not run:
# `sahmi_datasets` should be the output of all samples from `prep_dataset()`
slsd <- slsd(lapply(sahmi_datasets, `[[`, "kreport"))
real_taxids_slsd <- slsd$filter(
    pl$col("r1")$gt(0),
    pl$col("r2")$gt(0),
    pl$col("r3")$gt(0),
    pl$col("p1")$lt(0.05),
    pl$col("p2")$lt(0.05),
    pl$col("p3")$lt(0.05)
)$get_column("taxid")

## End(Not run)
```

---

taxa_counts *Quantitation of microbes*

Description

After identifying true taxa, reads assigned to those taxa are extracted and then undergo a series of filters. The cell barcode and UMI are used to demultiplex the reads and create a barcode x taxa counts matrix. The full taxonomic classification of all resulting barcodes and the number of counts assigned to each clade are tabulated.

Usage

```
taxa_counts(umi_list, samples = NULL)
```

Arguments

`umi_list`	A list of polars DataFrame for UMI data returned by prep_dataset.
`samples`	A character of sample identifier for each element in `umi_list`.

Value

A polars DataFrame.

See Also

https://github.com/sjdlabgroup/SAHMI

Examples

```
## Not run:
# `umi_list` should be the output of all samples from `prep_dataset()`, and
# filtered by `slsd()` and `blsd()`
taxa_counts(umi_list, basename(names(umi_list)))

## End(Not run)
```

Index