Package ‘MethylSeqR’

May 20, 2025

Type Package

Title Process and Analyze Nanopore Sequencing Methylation Data

Version 0.8.1

Description
MethylSeqR creates methylation databases and processes them with customizable options.
Data can be summarized by positions, windows, or provided genomic annotations (e.g., BED files).
The package includes quality control, differential methylation analysis, and sliding window support.

License MIT + file LICENSE

Encoding UTF-8

LazyData false

RoxygenNote 7.3.2

Imports arrow,
DBI,
dplyr,
dbplyr,
duckdb,
duckplyr,
ggplot2,
progress,
tidyr,
utils,
readr

Suggests knitr,
rmarkdown

VignetteBuilder knitr

Contents

`calc_ch3_diff` *Calculate Differential Methylation*

Description

This function calculates differential methylation between specified case and control groups using various statistical methods. The results are stored in a DuckDB database for further analysis.

Usage

```
calc_ch3_diff(
  ch3_db,
  call_type = "positions",
  cases,
  controls,
  mod_type = "mh",
  calc_type = "fast_fisher"
)
```

Arguments

`ch3_db`	A list containing the database file path. This should be a valid "ch3_db" class object.
`call_type`	A string representing the name of the table in the database from which to pull the data. Default is "positions".
`cases`	A character vector containing the sample names for the case group.
`controls`	A character vector containing the sample names for the control group.

`mod_type`	A string indicating the type of modification to analyze. Default is "mh" for methylation/hydroxymethylation.
`calc_type`	A string specifying the statistical method to use for calculating p-values. Options include "fast_fisher", "r_fisher", and "log_reg". Default is "fast_fisher".

Details

The function connects to the specified DuckDB database and retrieves methylation data from the specified call type table. It summarizes the data for cases and controls, calculates p-values based on the specified method, and stores the results in the "meth_diff" table.

Value

A list containing the updated "ch3_db" object with the latest tables in the database, including "meth_diff".

Examples

```
# Specify the path to the database
ch3_db <- system.file("my_data.ch3.db", package = "MethylSeqR")

# Get methylation statistics for the 'positions' call type without plotting
calc_ch3_diff(ch3_db = ch3_db,
              call_type = "positions",
              cases = c("Blood1_chr21", "Blood2_chr21", "Blood3_chr21"),
              controls = c("Sperm1_chr21", "Sperm2_chr21", "Sperm3_chr21")))
```

`calc_ch3_samplecor` *Correlation Matrix of Modified Sequence Data*

Description

This function calculates and optionally plots a correlation matrix for methylation or other modification fraction data from genomic positions. It supports position-based, region-based, and window-based calls and provides visualization using ggplot2.

Usage

```
calc_ch3_samplecor(
  ch3_db,
  call_type = c("positions"),
  plot = TRUE,
  save_path = NULL,
  plot_sample_order = NULL,
  plot_title = "Sample Correlation Matrix",
  max_rows = NULL
)
```

Arguments

`ch3_db`	A string. The path to the database containing ch3 files from nanopore data.
`call_type`	A character vector specifying the type of data to retrieve from the database. Options are "positions", "regions", or "windows". Default is "positions".
`plot`	A logical value. If TRUE, a correlation heatmap with correlation values is plotted. Default is FALSE.
`save_path`	A string. The file path to save the plot (e.g., .pdf or .png). If NULL, the plot is not saved. Default is NULL.
`plot_sample_order`	A character vector specifying the desired order of samples in the plot. Default is NULL, which uses the default order.
`plot_title`	A string. The title of the correlation heatmap. Default is "Sample Correlation Matrix".
`max_rows`	The maximum amount of rows wanted for calculation. This argument can help analysis run faster when there is a lot of data.

Details

This function connects to the ch3 files database, retrieves data based on the ‘call_type‘ parameter, and computes the Pearson correlation matrix for each sample. For "regions", data is aggregated by sample and region name. For "windows", data is aligned to common genomic windows before computing the correlation matrix. If ‘plot = TRUE‘, the function generates a ggplot2 heatmap with correlation values displayed. Optionally, the plot can be saved to a specified file path.

Value

A correlation matrix of modification fractions across samples. If ‘plot = TRUE‘, a ggplot object of the correlation matrix heatmap is also printed. If ‘save_path‘ is specified, the plot is saved to the given file path.

Note

The function assumes that the database contains tables named according to the ‘call_type‘ parameter (e.g., "positions", "regions", "windows"). It calculates the correlation matrix using the ‘pairwise.complete.obs‘ method, which handles missing data.

Examples

```
# Specify the path to the database
ch3_db <- system.file("my_data.ch3.db", package = "MethylSeqR")

# Run the correlation matrix function using the 'positions' call type and plot the results
calc_ch3_samplecor(ch3_db = ch3_db, call_type = "positions")
```

`classify_ch3_reads` *Classify Reads as Case or Control Based on Methylation Profiles*

Description

This function classifies reads in the ‘reads‘ table of a DuckDB database as either ‘"case"‘, ‘"control"‘, or ‘"unknown"‘ based on similarity to reference methylation fractions in a ‘key_table‘ (e.g., a collapsed windows file). Classification is based on how close the read’s ‘mh_frac‘ is to the average case or control ‘mh_frac‘, within a user-defined ‘meth_diff_threshold‘.

Usage

```
classify_ch3_reads(
  ch3_db,
  table_name = "classified_reads",
  reads_table = "reads",
  key_table,
  case,
  control,
  meth_diff_threshold = 0.1
)
```

Arguments

`ch3_db`	Path to a DuckDB ‘.db‘ file created by this package (e.g., from ‘summarize_reads()‘).
`table_name`	Character. Name of the output table to store classified reads (default: "classified_reads").
`reads_table`	Character. Name of the reads table in which you want to classify the reads (default: "reads").
`key_table`	Path to a CSV, TSV, or BED file generated by ‘collapse_windows()‘. Must include the columns: ‘chrom‘, ‘start‘, ‘end‘, ‘avg_mh_frac_control‘, ‘avg_mh_frac_case‘, and ‘avg_meth_diff‘.
`case`	Character string used to label case reads (e.g., ‘"case"‘).
`control`	Character string used to label control reads (e.g., ‘"control"‘).
`meth_diff_threshold`	Numeric value specifying the maximum difference in ‘mh_frac‘ allowed to match either case or control averages. Must be less than half the minimum absolute value of ‘avg_meth_diff‘ to prevent ambiguous classifications.

Details

This function runs entirely in SQL for scalability. It performs an interval join between the ‘reads‘ table and the key table on ‘chrom‘ and CpG position range, then classifies each read based on proximity of ‘mh_frac‘ to either ‘avg_mh_frac_control‘ or ‘avg_mh_frac_case‘.

Value

Invisibly returns the open database connection with a new table named ‘classified_reads‘ added to the database. This table includes:

- `sample_name` – Sample identifier
- `read_id` – Unique read identifier
- `first_cpg_pos` – First CpG position of the read
- `last_cpg_pos` – Last CpG position of the read
- `mh_frac` – Methylation fraction of the read
- `classification` – ‘"case"‘, ‘"control"‘, or ‘"unknown"‘

Examples

```
## Not run:
classify_ch3_reads(
  ch3_db = "my_data.ch3.db",
  key_table = "key_table.csv",
  case = "treated",
  control = "untreated",
  meth_diff_threshold = 0.1
)

## End(Not run)
```

`collapse_ch3_windows` *Collapse Windows Based on Methylation Differences*

Description

This function collapses significant windows in a methylation dataset by merging contiguous regions that meet the specified criteria. Can only collapse windows once a differential modification analysis (calc_ch3_diff()) has been called.

Usage

```
collapse_ch3_windows(
  ch3_db,
  table_name = "collapsed_windows",
  max_distance = 1000,
  sig_cutoff = 0.05,
  min_diff = 0.5
)
```

Arguments

`ch3_db`	A DuckDB database connection object or path to the database.
`table_name`	Character. Name of the output table to store collapsed windows (default: "collapsed_windows").
`max_distance`	Numeric. The maximum allowable distance between consecutive significant windows for merging (default: 1000).
`sig_cutoff`	Numeric. The significance threshold for adjusted p-values (default: 0.05).
`min_diff`	Numeric. The minimum absolute methylation difference required for inclusion in the analysis (default: 0.5).

Details

The function performs the following steps:

- Filters the ‘mod_diff_windows‘ to retain only significant windows where ‘p_adjust <= sig_cutoff‘ and ‘ABS(meth_diff) >= min_diff‘.
- Assigns a new region identifier based on proximity (‘max_distance‘) and the direction of methylation differences.
- Collapses regions by grouping contiguous windows, computing the average methylation difference (‘avg_meth_diff‘), and counting the number of merged windows.

Value

This function does not return an object; it creates or replaces the ‘collapsed_windows‘ table in the database.

`export_ch3_table` *Export Tables from the ch3 Database*

Description

This function exports specified tables from the ch3 database to CSV files. Can export one or multiple tables as a time. It checks whether each table exists in the database before exporting, and provides informative messages for any missing tables. The output CSV files are saved at the specified path.

Usage

```
export_ch3_table(ch3_db, table = "positions", out_path)
```

Arguments

`ch3_db`	A string. The path to the database containing ch3 files from nanopore data.
`table`	A character vector specifying the table to be exported from the database. Default is "positions".
`out_path`	A string. The path to the directory where the CSV files will be saved. The file will automatically be named "table name.csv".

Details

The function connects to the specified database and iterates through the list of table names provided in the ‘tables‘ parameter. For each table that exists in the database, it reads the table into R and writes it as a CSV file to the location specified by ‘out_path‘. If a table does not exist in the database, a message is printed indicating this.

In case of any error during the execution, a custom error message is displayed. The function ensures that the database connection is closed safely using the ‘finally‘ block.

Value

NULL. The function writes the specified tables to CSV files.

Note

The function assumes that the tables specified in ‘tables‘ exist in the database and can be accessed via the ‘DBI‘ package.

---

filter_ch3_table *Filter a DuckDB Table and Save to a New Table*

Description

Applies filter conditions to a table in a DuckDB database and saves the result to a new or replaced output table. This is done lazily using ‘dbplyr‘, so the filtering is translated to SQL and executed inside the database (not in R).

Usage

```
filter_ch3_table(ch3_db, input_table, output_table, ...)
```

Arguments

`ch3_db`	Path to the DuckDB database file (e.g., ‘"my_data.ch3.db"‘).
`input_table`	Name of the table to filter.
`output_table`	Name of the output table to create or overwrite with the filtered results.
`...`	Filtering expressions (e.g., ‘score > 0.5‘, ‘gene_id == "abc"‘). These are unquoted expressions passed directly to ‘dplyr::filter()‘.

Value

Invisibly returns the path to the DuckDB file (invisibly).

Examples

```
## Not run:
filter_ch3_table(ch3_db = train_db,
input_table = "collapsed_windows",
output_table = "collapsed_windows",
!(chrom %in% c("chrX", "chrY")))

## End(Not run)
```

get_ch3_cols *Get Column Names from a DuckDB Table*

Description

Returns a character vector of column names from a specified table in a DuckDB database file.

Usage

```
get_ch3_cols(ch3_db, table_name)
```

Arguments

`ch3_db`	Path to the DuckDB database file (e.g., ‘"my_data.ch3.db"‘).
`table_name`	Name of the table whose column names are to be retrieved.

Value

A character vector of column names from the specified table.

Examples

```
## Not run:
get_ch3_cols("my_data.ch3.db", "windows")

## End(Not run)
```

get_ch3_dbinfo *Get Database Statistics*

Description

Prints out a summary of the ch3 database, including size, tables, and unique sample names.

Usage

```
get_ch3_dbinfo(ch3_db)
```

Arguments

`ch3_db`	Path to the ‘.ch3.db‘ file or a ‘ch3_db‘ object.

Value

Invisibly returns a list of stats from the database.

Examples

```
# Specify the path to the database
ch3_db <- system.file("my_data.ch3.db", package = "MethylSeqR")

# Get database statistics
get_ch3_dbinfo(ch3_db = ch3_db)
```

`get_ch3_stats` *Compute Statistics for a Ch3 File*

Description

This function computes various statistics for a given Ch3 file stored in Parquet format, including total calls, total reads, mean read length, CpG coverage at different read thresholds, flag distributions, and high-quality call counts based on probability and base quality thresholds.

Usage

```
get_ch3_stats(
  ch3_file,
  log_file = NULL,
  min_reads = c(1, 5, 10, 15),
  call_prob_threshold = 0.9,
  base_qual_threshold = 10,
  silently = FALSE
)
```

Arguments

`ch3_file`	Character. Path to the Ch3 file in Parquet format.
`log_file`	Character (optional). Path to a log file where the output will be written. If NULL, results are printed to the console.
`min_reads`	Numeric vector. A set of thresholds for reporting CpG coverage at different minimum read counts. Default: `c(1, 5, 10, 15)`.
`call_prob_threshold`	Numeric. The minimum modification probability to consider a high-confidence call. Default: `0.9`.
`base_qual_threshold`	Numeric. The minimum base quality required to count as a high-quality call. Default: `10`.
`silently`	Logical. If `TRUE`, suppresses console output. Default: `FALSE`.

Details

The function reads the Ch3 file as a dataset and computes basic statistics about the calls and reads. It also evaluates CpG coverage based on different read count thresholds, distributions of flag values, and counts of high-quality calls based on user-defined probability and quality thresholds.

If a `log_file` is provided, the results are written to it. Otherwise, they are printed to the console unless `silently = TRUE`.

Value

A list containing:

`num_calls` Total number of modification calls in the file.

`num_reads` Total number of unique reads in the file.

`cpg_coverage` A matrix with CpG coverage counts at different `min_reads` thresholds.

`flag_counts` A data frame with the count and percentage of calls per flag value.

`high_conf_calls` A data frame with the count and percentage of calls with modification probability above `call_prob_threshold`.

`high_qual_calls` A data frame with the count and percentage of calls with base quality above `base_qual_threshold`.

`avg_read_length` The mean read length.

Examples

```
## Not run:
get_ch3_stats("example.ch3.parquet")
get_ch3_stats("example.ch3.parquet", log_file = "stats.log", silently = TRUE)

## End(Not run)
```

`get_ch3_table` *Collect Table from DuckDB Database as Tibble*

Description

This function connects to a DuckDB database and collects a specified table as a tibble.

Usage

```
get_ch3_table(ch3_db, table_name, max_rows = NULL)
```

Arguments

`ch3_db`	A list containing the database file path. This should be a valid "ch3_db" class object.
`table_name`	A string representing the name of the table to collect from the database.
`max_rows`	The maximum amount of rows wanted for calculation. This argument can help analysis run faster when there is a lot of data.

Details

The function establishes a connection to the DuckDB database using `.helper_connectDB`. It retrieves the specified table as a tibble. If an error occurs during table retrieval, a message with the error is displayed. The database connection is closed after retrieving the data, regardless of success or failure.

Value

A tibble containing the collected data from the specified database table. If the table retrieval fails, an empty tibble is returned.

Examples

```
# Assuming ch3_db is a valid database object and "positions" is a table in the database
ch3_db <- system.file("my_data.ch3.db", package = "MethylSeqR")
positions = get_ch3_table(ch3_db, "positions")
```

---

make_ch3_archive *Create a Compressed Methylation Call Archive (.ch3) from a calls file.*

Description

Reads a delimited methylation call file and writes it to a compressed archive format. Useful for standardizing and compressing methylation data across multiple samples.

Usage

```
make_ch3_archive(file_name, sample_name, out_path, short_ids = TRUE)
```

Arguments

`file_name`	Path to the input file (tab-delimited) containing methylation calls.
`sample_name`	Name of the sample to embed in the output archive filenames.
`out_path`	Output directory where the ‘.ch3‘ files will be written.
`short_ids`	Logical; default is ‘TRUE‘, shortens ‘read_id‘ by trimming prefixes to reduce file size.

Details

The input file should contain specific columns such as ‘read_id‘, ‘chrom‘, ‘ref_position‘, ‘ref_mod_strand‘, ‘query_kmer‘, ‘call_prob‘, ‘call_code‘, ‘base_qual‘, and ‘flag‘. The function filters out unplaced ’-’ strand reads with position 0, adds new fields like ‘start‘ and ‘end‘, and writes the result as a compressed Arrow dataset with ‘zstd‘ compression.

Value

(Invisibly) the path template to the output archive files. Also prints a message with timing information.

Examples

```
## Not run:
make_ch3_archive("calls.tsv", "sample1", "output/", short_ids = TRUE)

## End(Not run)
```

make_ch3_db *Create a CH3 Database from Parquet Files*

Description

This function processes a collection of CH3 files stored in Parquet format and creates a DuckDB database containing filtered and structured methylation call data.

Usage

```
make_ch3_db(
  ch3_files,
  db_name,
  chrom = NULL,
  min_read_length = 50,
  min_call_prob = 0.9,
  min_base_qual = 10,
  flag = NULL,
  chr_prefix = TRUE
)
```

Arguments

`ch3_files`	A character vector of file or directory paths containing CH3 Parquet files. If a directory is provided, all files within it will be processed.
`db_name`	A string representing the path where the CH3 database will be created. The extension ".ch3.db" will be appended if not already present.
`chrom`	An optional character string specifying a chromosome to filter the data. If NULL, all chromosomes will be included.
`min_read_length`	A numeric value specifying the minimum read length required for inclusion in the database. Default is 50.
`min_call_prob`	A numeric value representing the minimum call probability threshold. Only calls with a probability greater than or equal to this value will be included. Default is 0.9.
`min_base_qual`	A numeric value representing the minimum base quality threshold. Only reads with quality scores at or above this value will be included. Default is 30.
`flag`	An optional numeric value specifying a flag-based filter for the data. If NULL, no flag filtering is applied.
`chr_prefix`	A boolean value stating whether or not to keep the chr prefix in the chromosome column. Default is TRUE, and "chr" will be kept in front of every chr number.

Details

This function reads CH3 files stored in Parquet format and imports them into a DuckDB database. The data is filtered based on user-specified criteria, including chromosome, read length, call probability, base quality, and flag values. If a table already exists in the database, it will be dropped and recreated.

Value

A list of class ‘"ch3_db"‘, containing:

`db_file`	The path to the created DuckDB database file.
`tables`	A list of tables available in the database.

Examples

```
# Example usage
ch3_files <- system.file("new_test_data", package = "MethylSeqR")
db_name <- tempfile("example_ch3")

make_ch3_db(ch3_files, db_name)
```

`MethylSeqR` *MethylSeqR: Process and Analyze Nanopore Sequencing Methylation Data*

Description

MethylSeqR creates methylation databases and processes them with customizable options. Data can be summarized by positions, windows, or provided genomic annotations (e.g., BED files). The package includes quality control, differential methylation analysis, and sliding window support.

MethylSeqR is an R package developed by Wasatch Biolabs for efficient preprocessing, summarization, and visualization of methylation data. It supports native DNA methylation sequencing workflows and is optimized for high-throughput pipelines.

Details

This package is designed to support methylation analysis using Oxford Nanopore sequencing data. It builds efficient DuckDB-backed databases and supports downstream summary and visualization workflows.

Features

- Database-backed storage using DuckDB - Sliding window, positional, regional, and read-level summarization of methylation levels - Support for multiple methylation contexts - Quality Control Visualization - Efficient Differential Methylation Analysis

Author(s)

Jonathon Hill <`jonathon@wasatchbiolabs.com`> (aut, cre)
Hailey Zimmerman <`hailey@renewbt.com`> (aut)

Jonathon Hill <`jonathon@wasatchbiolabs.com`> (aut, cre) Hailey Zimmerman <`hailey@renewbt.com`> (aut)

References

Wasatch Biolabs (2025). *MethylSeqR: Tools for Methylation Analysis in Clinical and Research Settings.* https://www.wasatchbiolabs.com/

For bug reports and feature requests: https://github.com/Wasatch-Biolabs-Bfx/MethylSeqR

plot_ch3_cov *Calculate and Plot Coverage Statistics*

Description

This function calculates and optionally plots statistics for coverage data from methylation sequencing experiments. It can handle both positional and regional methylation data.

Usage

```
plot_ch3_cov(
  ch3_db,
  call_type = "positions",
  plot = TRUE,
  save_path = NULL,
  max_rows = NULL
)
```

Arguments

ch3_db	A data base either linking to the file name or of class ch3_db.
call_type	Either positions or regions data to analyze coverage on.
plot	Logical, if TRUE, the function will generate a histogram of the coverage data. Default is FALSE.
save_path	Pathway to save the plot to. Usually .pdf or .png.
max_rows	The maximum amount of rows wanted for calculation. This argument can help analysis run faster when there is a lot of data.

Value

If plot is FALSE, the function prints summary statistics and percentiles of the coverage data. If plot is TRUE, it prints a histogram of the log-transformed coverage data.

Examples

```
# Specify the path to the database
ch3_db <- system.file("my_data.ch3.db", package = "MethylSeqR")

# Get coverage statistics for the 'positions' call type without plotting
plot_ch3_cov(ch3_db = ch3_db, call_type = "positions")
```

`plot_ch3_modfrac` *Get Methylation Statistics from the ch3 Database*

Description

This function retrieves and calculates methylation statistics (mean methylation fractions) from a specified table in the ch3 database. It can either return summary statistics or plot a histogram of the methylation values, depending on the user's preference.

Usage

```
plot_ch3_modfrac(
  ch3_db,
  call_type = c("positions", "regions"),
  plot = TRUE,
  save_path = NULL,
  max_rows = NULL
)
```

Arguments

`ch3_db`	A string. The path to the database containing ch3 files from nanopore data.
`call_type`	A character vector specifying the type of data to retrieve from the database. Default is "positions". Can also be "regions".
`plot`	A logical value. If TRUE, a histogram of methylation values is plotted. Default is FALSE.
`save_path`	Pathway to save the plot to. Usually .pdf or .png.
`max_rows`	The maximum amount of rows wanted for calculation. This argument can help analysis run faster when there is a lot of data.

Details

The function connects to the specified database, checks for the existence of the relevant table, and retrieves methylation fraction data. If the table contains methylation data, it prioritizes the 'mh_frac' column over others. Depending on the 'call_type', it can compute statistics for either per base or per region. If 'plot' is set to TRUE, it generates a histogram of the methylation values.

Value

If 'plot' is FALSE, the function prints a summary of the methylation statistics and quantiles. If 'plot' is TRUE, it displays a histogram of methylation values.

Note

The function assumes that the database has tables named according to the 'call_type' parameter (e.g., "positions", "regions"). It also expects specific columns for methylation data to exist.

Examples

```
# Specify the path to the database
ch3_db <- system.file("my_data.ch3.db", package = "MethylSeqR")

# Get methylation statistics for the 'positions' call type without plotting
plot_ch3_modfrac(ch3_db = ch3_db, call_type = "positions")
```

`plot_ch3_pca` *Perform PCA on Methylation Data*

Description

This function performs Principal Component Analysis (PCA) on methylation data retrieved from a DuckDB database. It aggregates the methylation fraction data based on the specified call type and prepares it for PCA analysis.

Usage

```
plot_ch3_pca(
  ch3_db,
  call_type = "positions",
  save_path = NULL,
  max_rows = NULL
)
```

Arguments

`ch3_db`	A list containing the database file path. This should be a valid "ch3_db" class object.
`call_type`	A string representing the name of the table in the database from which to pull the data. Default is "positions".
`save_path`	Pathway to save the plot to. Usually .pdf or .png.
`max_rows`	The maximum amount of rows wanted for calculation. This argument can help analysis run faster when there is a lot of data.

Details

The function connects to the specified DuckDB database, retrieves the methylation data from the specified call type table, and aggregates the data either by regions or chromosomal positions. PCA is then performed on the aggregated data, and a scatter plot of the first two principal components is generated.

Value

A PCA plot is produced, showing the first two principal components of the methylation data. The function also prints a summary of the PCA results and the PCA data used for plotting.

Examples

```
# Specify the path to the database
ch3_db <- system.file("my_data.ch3.db", package = "MethylSeqR")

# Calculate PCA
plot_ch3_pca(ch3_db)
```

`print.ch3_db` *Helper Function to Print Database Tables*

Description

This function connects to a DuckDB database and prints the names of all tables currently in the database. It can also print the first few rows of specific tables.

Usage

```
## S3 method for class 'ch3_db'
print(ch3_db)
```

Arguments

`ch3_db`	A character string or an object of class ch3_db representing the DuckDB database to connect to.
`tables`	A character vector specifying the names of tables to print. Defaults to the last table given, which will print all tables. If specific table names are provided, it will only print those (ex. "positions", "regions", "meth_diff"). If tables ="all", the function will print out all table sin the database...

Details

The function establishes a connection to the database, retrieves the list of tables, and prints their names. If the specified table exists, it also prints the first few rows of that table. If a specified table does not exist, a message is printed to indicate this.

Value

None. This function is called for its side effects (printing information).

Examples

```
# Specify the path to the database
 ch3_files <- system.file("test_data", package = "MethylSeqR")
 ch3_db <- tempfile("example_db")

 # Print out tables in the database
 make_pos_db(ch3_files, ch3_db) |> print()
```

`remove_ch3_table` *Remove a table from a database connection.*

Description

This function checks if a table exists and, if so, removes it. It provides messages about the outcome.

Usage

```
remove_ch3_table(ch3_db, table_name)
```

Arguments

`ch3_db`	A database object.
`table_name`	A string specifying the name of the table to remove.

Value

Invisibly returns the database with the removed table.

`run_ch3_analysis` *Run Differential Analysis on Methylation Data*

Description

This function performs a differential analysis on methylation data based on the specified call type and applies the appropriate summarization method. It supports analysis of positions, regions, or windows. The function handles the summarization of methylation data and performs differential modification analysis based on case-control comparisons.

Usage

```
run_ch3_analysis(
  ch3_db,
  out_path,
  call_type,
  region_file = NULL,
  window_size = 1000,
  step_size = 10,
  cases,
  controls,
  mod_type = "mh",
  calc_type = "fast_fisher",
  p_val_max = 0.05
)
```

Arguments

`ch3_db`	A 'ch3_db' object representing the DuckDB database containing methylation data. The database should include necessary tables for the analysis, such as positions, regions, or windows.
`out_path`	The directory in which the "Mod_Diff_Analysis_Results" directory containing result data will be written out too. If the user does not provide a directory, the working directory will be used.
`call_type`	A character string specifying the type of data to analyze. Must be one of: `"positions"`, `"regions"`, or `"windows"`. This determines the summarization approach to use.
`region_file`	A character string specifying the path to the region annotation file (required if 'call_type' is '"regions"'). This file should be in a supported format (e.g., BED, CSV, TSV).
`window_size`	An integer specifying the window size for summarizing methylation data if 'call_type' is '"windows"'. The default value is 1000.
`step_size`	An integer specifying the step size for sliding windows if 'call_type' is '"windows"'. The default value is 10.
`cases`	A character vector of sample names to be used as cases in the differential analysis. This argument is required and cannot be NULL.
`controls`	A character vector of sample names to be used as controls in the differential analysis. This argument is required and cannot be NULL.
`mod_type`	A character string specifying the modification type to analyze. The default is '"mh"', which includes both methylation and hydroxymethylation. Other options are '"c"' for unmodified cytosine, '"m"' for methylation, and '"h"' for hydroxymethylation.
`calc_type`	A character string specifying the type of statistical test to use for the differential analysis. The default is '"fast_fisher"', but other calculation methods can be implemented.
`p_val_max`	The p value threshold in which significant differentially modified positions/regions/windows will be written out in the final directory.

Details

This function first summarizes the methylation data by the specified call type (positions, regions, or windows). It then proceeds with a differential modification analysis between the provided case and control samples. The analysis is tailored based on the selected modification type ('mod_type') and calculation method ('calc_type').

Value

The result of the differential analysis, typically in the form of a table or data frame with calculated statistics and p-values for each position, region, or window, depending on the 'call_type'. The result is printed to the console.

run_ch3_dplyr	*Execute an Expression on a DuckDB Table with Optional Materialization*

Description

Connects to a DuckDB database, evaluates a user-supplied expression on a specified table, and either collects the result into R or computes and stores it as a new table in the database.

Usage

```
run_ch3_dplyr(
  ch3_db,
  table_name,
  expr,
  mode = c("collect", "compute"),
  output_table = NULL
)
```

Arguments

ch3_db	Path to the DuckDB database file (e.g., ‘"my_data.ch3.db"‘).
table_name	Name of the table in the database to operate on.
expr	A function taking one argument (‘tbl_ref‘) and returning a lazy ‘dplyr‘ expression. The table reference (‘tbl_ref‘) will be passed as a ‘tbl()‘ object connected to the database.
mode	One of ‘"collect"‘ (default) or ‘"compute"‘. If ‘"collect"‘, the result is returned as a data frame in R. If ‘"compute"‘, the result is stored as a new table in the database.
output_table	Required if ‘mode = "compute"‘. Name of the output table to create or overwrite with the result of ‘expr(tbl_ref)‘.

Value

If ‘mode = "collect"‘, returns a data frame. If ‘mode = "compute"‘, returns ‘NULL‘ invisibly.

Examples

```
## Not run:
# Collect results of a filtered table into R
run_ch3_dplyr(
  ch3_db = "my_data.ch3.db",
  table_name = "methylation_data",
  expr = function(tbl_ref) dplyr::filter(tbl_ref, score > 0.5),
  mode = "collect"
)

# Store the filtered result in a new table inside the database
run_ch3_dplyr(
  ch3_db = "my_data.ch3.db",
  table_name = "methylation_data",
  expr = function(tbl_ref) dplyr::filter(tbl_ref, score > 0.5),
  mode = "compute",
  output_table = "filtered_data"
)

## End(Not run)
```

`run_ch3_qc` *Quality Control Wrapper for Methylation Data*

Description

This function serves as a wrapper for various quality control analyses on methylation data. It sequentially calculates coverage statistics, modification statistics, correlation analysis, and performs principal component analysis (PCA).

Usage

```
run_ch3_qc(ch3_db, call_type = "positions", plot = TRUE, max_rows = NULL)
```

Arguments

`ch3_db`	A database connection or object containing methylation data.
`call_type`	A character string indicating the type of call to retrieve data (e.g., "positions", "regions").
`plot`	A logical value indicating whether to generate plots for the statistical analyses. Defaults to TRUE. Set to FALSE to suppress plots.
`max_rows`	The maximum amount of rows wanted for calculation. This argument can help analysis run faster when there is a lot of data.

Details

The function first calculates coverage statistics using ‘get_cov_stats()‘, then computes modification statistics with ‘get_mod_stats()‘, follows up with correlation analysis via ‘cor_modseq()‘, and finally runs principal component analysis with ‘pca_modseq()‘.

Value

This function does not return a value. It performs calculations and may produce plots based on the results of the analysis.

Examples

```
# Specify the path to the database
ch3_db <- system.file("my_data.ch3.db", package = "MethylSeqR")

# Run quality control wrapper
run_qc(ch3_db, call_type = "positions")
```

run_ch3_sql *Execute a query on a CH3 Database*

Description

Connects to a DuckDB database, evaluates a user-supplied sql query and closes the database.

Usage

```
run_ch3_sql(ch3_db, query)
```

Arguments

`ch3_db`	Path to the DuckDB database file (e.g., ‘"my_data.ch3.db"‘).
`query`	An sql query supported by the duckdb database framework.

Value

a ch3_db object to allow piping

Examples

```
## Not run:
# Count the number of rows in the calls table
run_ch3_sql(
  ch3_db = "my_data.ch3.db",
  query = "CREATE TABLE call_count AS SELECT COUNT(*) AS num_rows FROM calls;")

## End(Not run)
```

summarize_ch3_positions *Summarize Methylation Positions from Calls Table*

Description

This function processes methylation call data from a DuckDB database, summarizing coverage and call counts into a ‘positions‘ table.

Usage

```
summarize_ch3_positions(
  ch3_db,
  table_name = "positions",
  mod_type = c("c", "m", "h", "mh"),
 chrs = c(as.character(1:22), paste0("chr", 1:22), "chrX", "chrY", "chrM", paste0("Chr",
    1:22), "ChrX", "ChrY", "ChrM"),
  min_num_calls = 1
)
```

Arguments

`ch3_db`	A DuckDB database connection or file path (character) to the ‘.ch3.db‘ file.
`table_name`	A string specifying what the user would like the name to be called in the database. Default is "positions".
`mod_type`	A character vector specifying the modification types to include. Options are ‘"c"‘ (unmodified cytosine), ‘"m"‘ (methylation), ‘"h"‘ (hydroxymethylation), and ‘"mh"‘ (methylated + hydroxymethylated).
`chrs`	A character vector specifying which chromosomes to include. Default includes all autosomes, sex chromosomes (chrX, chrY), and mitochondrial chromosome (chrM).
`min_num_calls`	Minimum number of calls required to include a position in the summary. Default is 1.

Details

The function: - Connects to the DuckDB database. - Summarizes methylation calls by ‘sample_name‘, ‘chrom‘, ‘start‘, and ‘end‘. - Computes coverage and call counts (‘m‘, ‘h‘, ‘mh‘, ‘c‘). - Filters based on ‘min_num_calls‘. - Creates a ‘positions‘ table in the database.

A progress bar is displayed during execution.

Value

The modified ‘ch3_db‘ object with the updated ‘positions‘ table.

Examples

```
# Specify the path to the database
ch3_db <- system.file("my_data.ch3.db", package = "MethylSeqR")

# Summarize Positions
summarize_ch3_positions(ch3_db)
```

`summarize_ch3_reads` *Summarize Reads in a Database*

Description

This function summarizes reads from a database, filtering and processing the data based on a provided key table (if given). It computes statistics on the reads such as the number of calls, CpG positions, and fractions of methylation (‘m‘), hemi-methylation (‘h‘), and total calls. The function interacts with the database to generate a ‘reads‘ table.

Usage

```
summarize_ch3_reads(
  ch3_db,
  table_name = "reads",
  key_table = NULL,
  min_CGs = 5
)
```

Arguments

`ch3_db`	A character string specifying the path to the DuckDB database.
`table_name`	A string specifying what the user would like the name to be called in the database. Default is "reads".
`key_table`	(Optional) A character string specifying the path to a key table (CSV, TSV, or BED) used for filtering reads. If NULL, no filtering is applied.
`min_CGs`	An integer specifying the minimum number of CG sites required for a read to be included in the summary.

Details

The function connects to the provided DuckDB database, optionally filters reads based on the key table, and then summarizes the read data. It creates a temporary table for the filtered reads (if a key table is provided) and creates a summary table called ‘reads‘ with information on the total number of calls, the positions of the first and last CG sites, and counts for different types of calls (‘m‘, ‘h‘, and ‘-‘).

Value

Invisibly returns the database object. The function also outputs a success message and the first few rows of the summarized ‘reads‘ table.

Examples

```
#Specify the path to the database
 ch3_db <- system.file("my_data.ch3.db", package = "MethylSeqR")
 region_bed = system.file("Islands_hg38_test.csv", package = "MethylSeqR")

 # Summarize Reads
 summarize_ch3_reads(ch3_db, region_bed)
```

`summarize_ch3_regions` *Summarize Methylation Data by Regions*

Description

This function summarizes methylation data from a DuckDB database based on specified regions defined in a BED, TSV, or CSV file. It performs a join operation between the methylation data and the regions specified in the annotation file, allowing for different types of joins.

Usage

```
summarize_ch3_regions(
  ch3_db,
  table_name = "regions",
  region_file,
  mod_type = c("c", "m", "h", "mh"),
 chrs = c(as.character(1:22), paste0("chr", 1:22), "chrX", "chrY", "chrM", paste0("Chr",
    1:22), "ChrX", "ChrY", "ChrM"),
  min_num_calls = 1
)
```

Arguments

`ch3_db`	A list containing the database file path. This should be a valid "ch3_db" class object.
`table_name`	A string specifying what the user would like the name to be called in the database. Default is "regions".
`region_file`	A string representing the path to the BED or CSV file that contains the region annotations.
`mod_type`	A character vector specifying the modification types to include. Options are ‘"c"‘ (unmodified cytosine), ‘"m"‘ (methylation), ‘"h"‘ (hydroxymethylation), and ‘"mh"‘ (methylated + hydroxymethylated).
`chrs`	A character vector specifying which chromosomes to include. Default includes all autosomes, sex chromosomes (chrX, chrY), and mitochondrial chromosome (chrM).
`min_num_calls`	An integer specifying the minimum number of calls required for inclusion in the summary. Default is 1.

Details

The function reads the region annotations from the regional annotation file and checks for its validity. It connects to the DuckDB database, creates a summarized table of methylation data based on the specified regions, and performs the join operation according to the specified join type. A progress bar is displayed during the summarization process. The resulting data is stored in a table called "regions" within the database.

Value

The updated ‘ch3_db‘ object with the summarized regions data added to the DuckDB database.

Examples

```
# Specify the path to the database
ch3_db <- system.file("my_data.ch3.db", package = "MethylSeqR")
region_bed = system.file("Islands_hg38_test.csv", package = "MethylSeqR")

# Summarize Regions using annotation table
summarize_ch3_regions(ch3_db, region_bed)
```

`summarize_ch3_windows` *Summarize Methylation Data using a Sliding Window*

Description

This function summarizes methylation data from a DuckDB database by creating sliding windows over the specified genomic regions. It allows for the adjustment of window size and step size to control the granularity of the summarization.

Usage

```
summarize_ch3_windows(
  ch3_db,
  table_name = "windows",
  window_size = 1000,
  step_size = 10,
  mod_type = c("c", "m", "h", "mh"),
 chrs = c(as.character(1:22), paste0("chr", 1:22), "chrX", "chrY", "chrM", paste0("Chr",
   1:22), "ChrX", "ChrY", "ChrM"),
  min_num_calls = 1,
  overwrite = TRUE
)
```

Arguments

`ch3_db`	A list containing the database file path. This should be a valid "ch3_db" class object.
`table_name`	A string specifying what the user would like the name to be called in the database. Default is "windows".
`window_size`	An integer specifying the size of the sliding window in base pairs. Default is 1000.
`step_size`	An integer specifying the number of base pairs to step forward with each window. Default is 10.
`mod_type`	A character vector specifying the modification types to include. Options are ‘"c"‘ (unmodified cytosine), ‘"m"‘ (methylation), ‘"h"‘ (hydroxymethylation), and ‘"mh"‘ (methylated + hydroxymethylated).
`chrs`	A character vector specifying which chromosomes to include. Default includes all autosomes, sex chromosomes (chrX, chrY), and mitochondrial chromosome (chrM).
`overwrite`	A logical indicating whether to overwrite the existing "windows" table if it exists. Default is TRUE.
`call_type`	A string indicating the type of data to summarize. Default is "positions".

Details

The function connects to a DuckDB database and removes any existing "windows" and "temp_table" tables if necessary. It creates a sequence of offsets based on the specified window and step sizes, and then it iterates through these offsets to generate sliding windows of methylation data. A progress bar is displayed during the operation.

The function utilizes the helper function ‘.make_window‘ to perform the actual window calculation and summarization. The resulting summarized data is stored in a table called "windows" within the database.

Value

The updated ‘ch3_db‘ object with the summarized windows data added to the DuckDB database.

Examples

```
 # Specify the path to the database
ch3_db <- system.file("my_data.ch3.db", package = "MethylSeqR")

# Summarize Windows
summarize_ch3_windows(ch3_db, window_size = 100, step_size = 100)
```

Index