Package 'shinyWGD'

November 13, 2024

Type Package

Title 'Shiny' Application for Whole Genome Duplication Analysis

Version 1.0.0

Maintainer Jia Li <li081766@gmail.com>

Description Provides a comprehensive 'Shiny' application for analyzing
Whole Genome Duplication ('WGD') events.
This package provides a user-friendly 'Shiny' web application for
non-experienced researchers to prepare input data and execute command lines
for several well-known 'WGD' analysis tools, including 'wgd', 'ksrates',
'i-ADHoRe', 'OrthoFinder', and 'Whale'. This package also provides the source code
for experienced researchers to adjust and install the package to their own server.
Key Features
1) Input Data Preparation
This package allows users to conveniently upload and format their data,
making it compatible with various 'WGD' analysis tools.
2) Command Line Generation
This package automatically generates the necessary command lines for
selected 'WGD' analysis tools, reducing manual errors and saving time.
3) Visualization
This package offers interactive visualizations to explore and interpret 'WGD' results,
facilitating in-depth 'WGD' analysis.
4) Comparative Genomics
Users can study and compare 'WGD' events across different species,
aiding in evolutionary and comparative genomics studies.
5) User-Friendly Interface
This 'Shiny' web application provides an intuitive and accessible interface,
making 'WGD' analysis accessible to researchers and 'bioinformaticians' of all levels.

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.3

SystemRequirements pandoc (>= 1.12.3), pandoc-citeproc

Imports shiny, shinyalert, stringr, vroom, fs, tidyr, data.table,
dplyr, ape, ks, mclust, htmltools, seqinr, httr, jsonlite

Suggests tidyverse, knitr, rmarkdown, DT, argparse, bslib, bsplus, english, fontawesome, igraph, shinyBS, shinyFiles, shinyWidgets, shinyjs, stringi, tools, testthat (>= 3.0.0)

NeedsCompilation no

Author Jia Li [aut, cre], Zhen Li [ctb], Arthur Zwaenepoel [ctb]

Repository CRAN

Date/Publication 2024-11-13 15:10:01 UTC

Contents

`analysisEachCluster` *Perform synteny analysis for identified clusters*

Description

This function performs synteny analysis for clusters identified by hierarchical clustering.

Usage

```
analysisEachCluster(
  segmented_file,
  segmented_anchorpoints_file,
  genes_file,
  cluster_info_file,
  identified_cluster_file,
  hcheight = 0.3
)
```

Arguments

segmented_file	The path to the segmented chromosome file.
segmented_anchorpoints_file	The path to the segmented anchorpoints file.
genes_file	genes.txt created by i-ADHoRe.
cluster_info_file	The path to the clustering information file.
identified_cluster_file	The path to the output file for identified clusters.
hcheight	The cutoff height for cluster identification (default: 0.3).

Value

A list containing information about identified clusters and their p-values.

---

bootStrapPeaks *Bootstrap Peaks in the Ks Distribution*

Description

This function performs bootstrapping on a given Ks (synonymous substitution rates) distribution to estimate peaks within the distribution.

Usage

```
bootStrapPeaks(
  ksRaw,
  binWidth = 0.1,
  maxK = 5,
  m = 3,
  peak.index = 1,
  peak.maxK = 2,
  spar = 0.25,
  rep = 1000,
  from = 0,
  to = maxK
)
```

Arguments

ksRaw	A numeric vector representing the raw Ks distribution to be bootstrapped.
binWidth	A numeric value indicating the bin width for histogram calculation.
maxK	A numeric value indicating the maximum Ks value to consider in the distribution.
m	An integer specifying the parameter for peak detection.

peak.index	An integer indicating the index of the peak to be estimated.
peak.maxK	A numeric value indicating the maximum Ks value for peak estimation.
spar	A numeric value controlling the smoothness of spline fitting.
rep	An integer specifying the number of bootstrap repetitions.
from	A numeric value indicating the lower bound for peak estimation.
to	A numeric value indicating the upper bound for peak estimation.

Value

A numeric vector containing bootstrapped peak estimates.

`calculateKsDistribution4wgd_multiple`
Calculate the Ks Distribution for Multiple Speices

Description

This function takes a list of data files, calculates the Ks distribution, and returns the results.

Usage

```
calculateKsDistribution4wgd_multiple(
  files_list,
  binWidth = 0.1,
  maxK = 5,
  plot.mode = "weighted",
  include.outliers = FALSE,
  minK = 0,
  minAlnLen = 0,
  minIdn = 0,
  minCov = 0
)
```

Arguments

files_list	A list of file paths containing Ks data.
binWidth	The width of Ks bins for the distribution.
maxK	The maximum Ks value to consider.
plot.mode	The mode for plotting ("weighted", "average", "min", or "pairwise").
include.outliers	Whether to include outliers in the calculation.
minK	The minimum Ks value to include in the distribution.
minAlnLen	The minimum alignment length to include in the distribution.
minIdn	The minimum alignment identity to include in the distribution.
minCov	The minimum alignment coverage to include in the distribution.

Value

A list containing two data frames: "bar" for Ks distribution and "density" for density data.

CalHomoConcentration *Compute the -log10 of Poisson Distribution*

Description

This function calculates the -log10 of the p-value of a Poisson distribution given the parameters.

Usage

```
CalHomoConcentration(m, n, q, k)
```

Arguments

m	The total number of trials.
n	The total number of possible outcomes.
q	The observed number of successful outcomes.
k	The expected number of successful outcomes.

Value

The -log10 of the p-value.

CalPvalue *Compute the P-value of a Cluster using the Poisson Distribution*

Description

This function computes the P-value of a cluster using the Poisson distribution.

Usage

```
CalPvalue(m, n, q, k)
```

Arguments

m	The total number of all anchored points.
n	The product of the remapped gene number of the query species and subject species.
q	The number of anchored points in the cluster.
k	The product of the remapped gene number of the segmented chromosomes of the query species and subject species.

Value

The computed P-value.

`checkFileExistence` *Check File Existence in a Data Table*

Description

This function checks the existence of files specified in a data table.

Usage

```
checkFileExistence(data_table, working_wd)
```

Arguments

`data_table`	A data table with file paths in columns V2 and V3.
`working_wd`	A path of the working directory

Value

This function has no return value. It prints messages to the console.

`check_gff_from_file` *Check and Process GFF Input File from a Specific Path*

Description

This function checks the type of GFF input file specified by its path and processes it accordingly.

Usage

```
check_gff_from_file(gff_input_name, gff_input_path, working_wd)
```

Arguments

`gff_input_name`	The informal name of the GFF input file.
`gff_input_path`	The path to the GFF input file.
`working_wd`	A character string specifying the working directory to be used.

Value

A string containing the processed GFF file's path.

check_gff_input *Check and Prepare GFF/GTF Input File*

Description

This function checks the file format of a GFF/GTF input file and prepares it for analysis. It can handle both uncompressed and compressed formats.

Usage

```
check_gff_input(gff_input_name, gff_input_path, working_wd)
```

Arguments

gff_input_name	A descriptive name for the GFF/GTF file.
gff_input_path	The file path to the GFF/GTF file.
working_wd	A character string specifying the working directory to be used.

Value

The path to the prepared GFF file for analysis.

---

check_proteome_from_file
Check and Process Proteome Input File From a Special Path

Description

This function checks the type of proteome input file and processes it accordingly.

Usage

```
check_proteome_from_file(proteome_name, proteome_input, working_wd)
```

Arguments

proteome_name	The informal name of the proteome input file.
proteome_input	The proteome input data.
working_wd	A character string specifying the working directory to be used.

Value

A string containing the processed proteome file's path.

check_proteome_input *Check and Process Proteome Input File*

Description

This function checks the type of proteome input file and processes it accordingly.

Usage

```
check_proteome_input(proteome_name, proteome_input, working_wd)
```

Arguments

proteome_name	The informal name of the proteome input file.
proteome_input	The proteome input data.
working_wd	A character string specifying the working directory to be used.

Value

A string containing the processed proteome file's path.

cluster_synteny *Cluster Synteny Data and Generate Trees*

Description

This function clusters synteny data based on calculated p-values and generates trees for both column-based and row-based clustering. It then saves the cluster information and trees to output files.

Usage

```
cluster_synteny(
  segmented_file,
  segmented_anchorpoints_file,
  genes_file,
  out_file
)
```

Arguments

segmented_file	A character string specifying the file path for segmented data.
segmented_anchorpoints_file	A character string specifying the file path for segmented anchorpoints.
genes_file	A character string specifying the file path for genes information created by i-ADHoRe.
out_file	A character string specifying the output file path for saving cluster information.

Value

NULL (output files are generated with the specified information).

`computing_depth` *Compute the Depth of Anchored Points*

Description

This function calculates the depth of anchored points based on the provided parameters.

Usage

```
computing_depth(
  anchorpoint_ks_file,
  multiplicon_id,
  selected_query_chr,
  selected_subject_chr = NULL
)
```

Arguments

`anchorpoint_ks_file`	The file containing anchorpoint and Ks data.
`multiplicon_id`	The ID of the multiplicon to consider.
`selected_query_chr`	A list of selected query chromosomes.
`selected_subject_chr`	A list of selected subject chromosomes (optional).

Value

A list containing depth data frames, including "query_depth" and "subject_depth" if subject chromosomes are specified, or "depth" if not.

`computing_depth_paranome` *Compute the Depth of Anchored Points in a Paranome Comparison*

Description

This function computes the depth of anchored points in a paranome comparison based on the provided parameters.

Usage

```
computing_depth_paranome(
  anchorpoint_ks_file,
  multiplicon_id,
  selected_query_chr
)
```

Arguments

anchorpoint_ks_file	The file containing anchor point and Ks value data.
multiplicon_id	The IDs of the multiplicons to consider.
selected_query_chr	The list of selected query chromosomes.

Value

A list containing the depth dataframe.

CountOrthologs *Count Ortholog Genes in a Species*

Description

This function counts ortholog genes in a given species based on input data.

Usage

```
CountOrthologs(atomic.df, species)
```

Arguments

atomic.df	A data frame containing information about ortholog genes. It should have the following columns: - multiplicon: The multiplicon identifier. - geneX: The gene identifier in speciesX. - speciesX: The species name for geneX. - listX: The chromosome or list identifier for geneX. - coordX: The coordinate information for geneX. - geneY: The gene identifier in speciesY. - speciesY: The species name for geneY. - listY: The chromosome or list identifier for geneY. - coordY: The coordinate information for geneY. - level: The orthology level. - num_anchors: The number of anchors. - is_real: A flag indicating if the data is real. - Ks: The Ks value.
species	The species for which ortholog gene counts should be computed.

Value

A data frame summarizing the counts of ortholog genes for each chromosome.

create_ksrates_cmd *Create Ksrates Command Files from Shiny Input*

Description

Create Ksrates Command Files from Shiny Input

Usage

```
create_ksrates_cmd(input, ksratesconf, cmd_file)
```

Arguments

input	The Input object of Shiny.
ksratesconf	The path to the Ksrates configuration file.
cmd_file	The path to the main Ksrates command file to be generated.

create_ksrates_cmd_from_table *Create Ksrates Command Files from Data Table*

Description

This function generates command files for running Ksrates and related analyses based on a data table and configuration file.

Usage

```
create_ksrates_cmd_from_table(data_table, ksratesconf, cmd_file, focal_species)
```

Arguments

data_table	The data table containing information about species.
ksratesconf	The path to the Ksrates configuration file.
cmd_file	The path to the main Ksrates command file to be generated.
focal_species	The name of the focal species.

create_ksrates_configure_file_based_on_table

Create Ksrates Configuration File Based on Data Table

Description

This function generates a Ksrates configuration file based on a data table and other parameters.

Usage

```
create_ksrates_configure_file_based_on_table(
  data_table,
  focal_species,
  newick_tree_file,
  ksrates_conf_file,
  species_info_file,
  working_wd
)
```

Arguments

data_table	The data table containing information about species, proteomes, and GFF files.
focal_species	The name of the focal species.
newick_tree_file	The path to the Newick tree file.
ksrates_conf_file	The path to the Ksrates configuration file to be generated.
species_info_file	The path to the species information file.
working_wd	A character string specifying the working directory to be used.

create_ksrates_configure_file_v2

Create Ksrates Configuration File

Description

This function generates a configuration file for the Ksrates pipeline based on Shiny input.

Usage

```
create_ksrates_configure_file_v2(input, ksrates_conf_file, species_info_file)
```

Arguments

input	The Input object of Shiny.
ksrates_conf_file	The path to the Ksrates configuration file.
species_info_file	The path to the species information file.

`create_ksrates_expert_parameter_file` *Create ksrates Expert Parameter File*

Description

Create ksrates Expert Parameter File

Usage

```
create_ksrates_expert_parameter_file(ksrates_expert_parameter_file)
```

Arguments

ksrates_expert_parameter_file	The file is used to store the ksrates expert parameter

---

`dfltBWrange` *dfltBWrange*

Description

This function computes the default bandwidth range for kernel density estimation.

Usage

```
dfltBWrange(x, tau)
```

Arguments

x	The input data, which can be a numeric vector or matrix.
tau	A parameter used in bandwidth calculation.

Value

A list of bandwidth ranges for each dimension of the input data.

dfltCounts *dfltCounts*

Description

This function bins the input data into a regular grid.

Usage

```
dfltCounts(
  x,
  gridsize = rep(64, NCOL(x)),
  h = rep(0, NCOL(x)),
  supp = 3.7,
  range.x,
  w
)
```

Arguments

x	The input data, which should be a numeric matrix.
gridsize	A vector specifying the number of bins along each dimension.
h	A vector specifying the bandwidth (smoothing parameter) along each dimension.
supp	A parameter for determining the range of the bins.
range.x	A list specifying the range of values for each dimension.
w	A vector of weights for the data points.

Value

A list containing the binned counts and the range of values for each dimension.

downloadButton_custom *Creating a Custom Download Button*

Description

Use this function to create a custom download button or link. When clicked, it will initiate a browser download. The filename and contents are specified by the corresponding downloadHandler() defined in the server function.

Usage

```
downloadButton_custom(
  outputId,
  label = "Download",
  class = NULL,
  status = "primary",
  ...,
  icon = shiny::icon("download")
)
```

Arguments

outputId	The name of the output slot that the downloadHandler is assigned to.
label	The label that should appear on the button.
class	Additional CSS classes to apply to the tag, if any. Default NULL.
status	The status of the button; default is "primary."
...	Other arguments to pass to the container tag function.
icon	An icon() to appear on the button; default is icon("download").

Value

An HTML tag to allow users to download the object.

---

drvkde *drvkde*

Description

Compute the mth derivative of a binned d-variate kernel density estimate based on grid counts.

Usage

```
drvkde(x, drv, bandwidth, gridsize, range.x, binned = FALSE, se = TRUE, w)
```

Arguments

x	The input data.
drv	The order of the derivative to compute.
bandwidth	The bandwidth (smoothing parameter) along each dimension.
gridsize	The size of the grid.
range.x	A list specifying the range of values for each dimension.
binned	A logical indicating whether the input data is already binned.
se	A logical indicating whether to compute standard errors.
w	A vector of weights for the data points.

Value

A list containing the estimated density or derivative, and optionally, standard errors.

`extractCluster` *Extract clusters based on specified scaffolds*

Description

This function extracts clusters based on the specified scaffolds for both query and subject species. It filters the data frames containing segment information and atomic anchorpoints to retain only the relevant clusters.

Usage

```
extractCluster(segs.df, atomic.df, scaf.bycol, scaf.byrow)
```

Arguments

`segs.df`	A data frame containing segment information.
`atomic.df`	A data frame containing atomic anchorpoints.
`scaf.bycol`	A character vector specifying scaffolds for the query species.
`scaf.byrow`	A character vector specifying scaffolds for the subject species.

Value

A list containing two data frames: "segs" for segment information and "atomic" for atomic anchorpoints.

`extract_first_part` *Extract the first part of a string by splitting it at tab characters.*

Description

This function takes a string and splits it at tab characters. It then returns the first part of the resulting character vector.

Usage

```
extract_first_part(name)
```

Arguments

`name`	The input string to be split.

Value

Returns the first part of the input string.

find_peaks *Find Peaks in a Numeric Vector*

Description

This function identifies peaks in a numeric vector by analyzing the shape of the curve.

Usage

```
find_peaks(x, m = 3)
```

Arguments

x	A numeric vector in which peaks will be identified.
m	An integer indicating the half-width of the neighborhood to consider when identifying peaks. A larger value of m makes peak detection less sensitive.

Value

A numeric vector containing the indices of the identified peaks in the input vector x.

generateKsDistribution *Generate the Ks Distribution*

Description

This function generates a Ks (synonymous substitution rates) distribution from raw Ks values.

Usage

```
generateKsDistribution(ksraw, speciesName = NULL, maxK = 5)
```

Arguments

ksraw	A numeric vector containing raw Ks values.
speciesName	(Optional) A character string specifying the species name associated with the Ks values.
maxK	A numeric value indicating the maximum Ks value to consider in the distribution.

Value

A numeric vector containing the binned Ks distribution.

generate_ksd *Generate Kernel Density Estimates (KDE) for Ks Distribution*

Description

This function generates Kernel Density Estimates (KDE) for the Ks (synonymous substitution rates) distribution.

Usage

```
generate_ksd(ks_df, bin_width = 0.01, maxK = 5)
```

Arguments

ks_df	A data frame containing Ks values.
bin_width	The width of each bin for KDE calculation.
maxK	The maximum Ks value for the distribution.

Value

A list containing the following components:

- Ks: A numeric vector representing the KDE values.
- bin_width: The width of each bin used for KDE calculation.
- maxK: The maximum Ks value for the distribution.

get_segments *Get Segmented Data from Anchorpoints and Ks Values*

Description

This function extracts segmented data from anchorpoints and Ks (synonymous substitution rate) values, based on specified criteria, and writes the results to output files.

Usage

```
get_segments(
  genes_file,
  anchors_ks_file,
  multiplicons_file,
  segmented_file,
  segmented_anchorpoints_file,
  num_anchors = 10
)
```

Arguments

genes_file	A character string specifying the file path for genes information created by i-ADHoRe.
anchors_ks_file	A character string specifying the file path for anchorpoints Ks values data.
multiplicons_file	A character string specifying the file path for multiplicons information created by i-ADHoRe.
segmented_file	A character string specifying the output file path for segmented data.
segmented_anchorpoints_file	A character string specifying the output file path for segmented anchorpoints.
num_anchors	An integer specifying the minimum number of anchorpoints required.

Value

NULL (output files are generated with the specified information).

---

is.ksv *Check if an object is of class "ksv"*

Description

This function checks if the provided object is of class "ksv."

Usage

```
is.ksv(x)

is.ksv(x)
```

Arguments

x	The object to be checked.

Value

Returns TRUE if the object is of class "ksv"; otherwise, returns FALSE.

is.not.null *Check if an Object is Not NULL*

Description

This function checks if an object is not NULL.

Usage

```
is.not.null(x)
```

Arguments

x	An R object to check.

Value

A logical value indicating whether the object is not NULL.

is_fasta_cds *Check if a file is in FASTA format with cds sequences.*

Description

This function checks whether a given file is in FASTA format with cds sequences.

Usage

```
is_fasta_cds(file_path)
```

Arguments

file_path	The path to the input file.

Value

TRUE if the file is in FASTA format with cds sequences, FALSE otherwise.

ks_mclust_v2 *ks_mclust_v2*

Description

A wrapper to run emmix modeling using the mclust package.

Usage

```
ks_mclust_v2(input_data)
```

Arguments

input_data	The input data for clustering and modeling.

Value

A data frame containing clustering and modeling results.

---

map_informal_name_to_latin_name
Map Informal Names to Latin Names

Description

This function reads information from an Excel file (XLS) containing columns "latin_name," "informal_name," and "gff." It extracts the "latin_name" and "informal_name" columns, performs some data manipulation, and returns a data frame with these two columns.

Usage

```
map_informal_name_to_latin_name(sp_gff_info_xls)
```

Arguments

sp_gff_info_xls	The path to the Excel file containing species information.

Value

A data frame with "latin_name" and "informal_name" columns.

mix_logNormal_Ks *Log-Normal mixturing analyses of a Ks distributions for the whole paranome*

Description

Log-Normal mixturing analyses of a Ks distributions for the whole paranome

Usage

```
mix_logNormal_Ks(ksv, G = 1:5, k.nstart = 500, maxK = 5)
```

Arguments

ksv	A ksv object.
G	An integer vector specifying the range of the mixtured components. A BIC is calculated for each component. The default is G=1:5. For a formal analysis, it is recommended to use 1:10.
k.nstart	How many random sets should be chosen in the k-means clustering. For a formal analysis, it is recommended to use 500.
maxK	Maximum Ks values used in the mixture modeling analysis.

Value

A data frame with seven variables.

modeFinder *modeFinder*

Description

Find the mode (peak) of a univariate distribution.

Usage

```
modeFinder(x, bw = 0.1, from = 0, to = 5)
```

Arguments

x	A numeric vector or a kernel density estimate (KDE).
bw	Bandwidth for the KDE. Default is 0.1.
from	Starting point for mode search. Default is 0.
to	Ending point for mode search. Default is 5.

Value

The mode (peak) of the distribution.

obtain_chromosome_length *obtain_chromosome_length*

Description

Process species information file and extract chromosome lengths and mRNA counts from GFF files.

Usage

```
obtain_chromosome_length(species_info_file)
```

Arguments

species_info_file	A character string specifying the path to the species information file.

Value

A list containing two data frames: len_df for chromosome lengths and num_df for mRNA counts.

obtain_chromosome_length_filter *obtain_chromosome_length_filter*

Description

Process a data frame containing species information and extract chromosome lengths and mRNA counts from GFF files.

Usage

```
obtain_chromosome_length_filter(species_info_df)
```

Arguments

species_info_df	A data frame containing species information with columns "sp," "cds," and "gff."

Value

A list containing two data frames: len_df for chromosome lengths and num_df for mRNA counts.

obtain_coordiantes_for_anchorpoints

Obtain coordinates for anchorpoints from GFF files

Description

This function takes a file containing anchorpoints, GFF files for two species, and species names, and retrieves the coordinates of anchorpoints and associated genes from the GFF files.

Usage

```
obtain_coordiantes_for_anchorpoints(
  anchorpoints,
  species1,
  gff_file1,
  out_file,
  species2 = NULL,
  gff_file2 = NULL
)
```

Arguments

anchorpoints	A file containing anchorpoints information with columns like gene_x, gene_y, and other relevant data.
species1	The name of the first species.
gff_file1	The path to the GFF file for the first species.
out_file	The output file where the results will be saved.
species2	(Optional) The name of the second species. Specify this parameter and gff_file2 if working with two species.
gff_file2	(Optional) The path to the GFF file for the second species.

Value

None. The function saves the results to the specified out_file.

obtain_coordiantes_for_anchorpoints_ks

Obtain Coordinates and Ks Values for Anchorpoints

Description

This function extracts coordinates and Ks (synonymous substitution rate) values for anchorpoints from input data and merges them into a single output file.

Usage

```
obtain_coordiantes_for_anchorpoints_ks(
  anchorpoints,
  anchorpoints_ks,
  genes_file,
  out_file,
  out_ks_file,
  species
)
```

Arguments

anchorpoints	A character string specifying the file path for anchorpoints data.
anchorpoints_ks	A character string specifying the file path for anchorpoints Ks values data.
genes_file	A character string specifying the file path for genes information.
out_file	A character string specifying the output file path for coordinates.
out_ks_file	A character string specifying the output file path for Ks values.
species	A character string specifying the species name.

Value

NULL (output files are generated with the specified information).

---

obtain_coordiantes_for_segments

Obtain coordinates for segments in a comparison

Description

This function retrieves the coordinates for segments in a comparison based on the provided parameters.

Usage

```
obtain_coordiantes_for_segments(
  seg_file,
  sp1,
  gff_file1,
  out_file,
  sp2 = NULL,
  gff_file2 = NULL
)
```

Arguments

seg_file	The file containing segment data.
sp1	The species name for the first genome.
gff_file1	The GFF file for the first genome.
out_file	The output file to store the merged position data.
sp2	The species name for the second genome (optional).
gff_file2	The GFF file for the second genome (optional).

Value

NULL (the results are saved in the output file).

---

obtain_coordinates_for_segments_multiple

Obtain Coordinates for Segments in Multiple Synteny Blocks

Description

This function extracts coordinates for segments within multiple synteny blocks based on input dataframes.

Usage

```
obtain_coordinates_for_segments_multiple(seg_df, gff_df, input, out_file)
```

Arguments

seg_df	A dataframe containing information about synteny segments.
gff_df	A dataframe containing GFF (General Feature Format) information.
input	A list containing input data, typically multiple synteny query chromosomes.
out_file	A character string specifying the output file path.

Value

A dataframe with coordinates for segments within multiple synteny blocks.

`obtain_mean_ks_for_each_multiplicon` *Compute the Mean of Ks values for Each Multiplicon*

Description

This function takes as input a multiplicon file, an anchorpoint file, Ks values, and other relevant information. It calculates the mean of Ks values for each multiplicon and associates them with the corresponding data.

Usage

```
obtain_mean_ks_for_each_multiplicon(
  multiplicon_file,
  anchorpoint_file,
  species1,
  ks_file,
  outfile,
  anchorpointout_file,
  species2 = NULL
)
```

Arguments

`multiplicon_file`	A file containing multiplicon information.
`anchorpoint_file`	A file containing anchorpoints information with columns like geneX, geneY, and other relevant data.
`species1`	The name of the first species.
`ks_file`	A file containing Ks values.
`outfile`	The output file where the results will be saved.
`anchorpointout_file`	The output file for anchorpoint data with Ks values.
`species2`	(Optional) The name of the second species. Specify this parameter and ks_file if working with two species.

Value

None. The function saves the results to the specified outfile and anchorpointout_file.

parse_EMMIX *Read the EMMIX output for a range of components*

Description

Read the EMMIX output for a range of components

Usage

```
parse_EMMIX(emmix.out, G = 1:3)
```

Arguments

emmix.out	The output file from EMMIX software.
G	An integer vector specifying the range of the mixture components. The default is G=1:3.

Value

A data frame with seven variables.

parse_one_EMMIX *Read the EMMIX output for a specify number of components*

Description

Read the EMMIX output for a specify number of components

Usage

```
parse_one_EMMIX(emmix.out, ncomponent = 3)
```

Arguments

emmix.out	The output file from EMMIX software.
ncomponent	Number of components to read from the file.

Value

A data frame with seven variables.

`PeaksInKsDistributionValues` *Find Peaks in the Ks Distribution*

Description

This function identifies peaks in a distribution of Ks (synonymous substitution rates) values.

Usage

```
PeaksInKsDistributionValues(
  ks,
  binWidth = 0.1,
  maxK = 5,
  m = 3,
  peak.maxK = 2,
  spar = 0.25
)
```

Arguments

`ks`	A numeric vector containing Ks values for which peaks will be identified.
`binWidth`	A numeric value specifying the bin width for creating the histogram.
`maxK`	A numeric value indicating the maximum Ks value to consider.
`m`	An integer indicating the half-width of the neighborhood to consider when identifying peaks. A larger value of `m` makes peak detection less sensitive.
`peak.maxK`	A numeric value specifying the maximum Ks value to consider when identifying peaks.
`spar`	A numeric value controlling the smoothness of the spline fit. Higher values make the fit smoother.

Value

A numeric vector containing the identified peaks in the Ks distribution.

`read.wgd_ksd` *Read the output file of wgd ksd*

Description

Read the output file of wgd ksd

Usage

```
read.wgd_ksd(
  file,
  include_outliers = FALSE,
  min_ks = 0,
  min_aln_len = 0,
  min_idn = 0,
  min_cov = 0
)
```

Arguments

file	The output file of wgd ksd
include_outliers	Include outliers or not, default FALSE.
min_ks	Minimum Ks value, default 0.
min_aln_len	Minimum alignment length, default 0.
min_idn	Minimum alignment identity, default 0.
min_cov	Minimum alignment coverage, default 0.

Value

A ksv object, which is a list including:

- ks_df: the data frame that used for following analysis
- ks_dist: a list including a vector of Ks values in the distribution
- raw_df: raw data
- filters: filters that applied to the raw data

---

read_data_file *Read Data from Uploaded File*

Description

This function reads data from an uploaded file in a Shiny application and returns it as a data frame.

Usage

```
read_data_file(uploadfile)
```

Arguments

uploadfile	The object representing the uploaded file obtained through the Shiny upload function.

Value

A data frame containing the data from the uploaded file.

`relativeRate` *relativeRate*

Description

Compute relative rates using input data files and statistical computations.

Usage

```
relativeRate(
  ksv2out_1_file,
  ksv2out_2_file,
  ksv_between_file,
  KsMax,
  low = 0.025,
  up = 0.975,
  bs = 1000
)
```

Arguments

`ksv2out_1_file`	A character string specifying the path to the first input data file.
`ksv2out_2_file`	A character string specifying the path to the second input data file.
`ksv_between_file`	A character string specifying the path to the third input data file.
`KsMax`	A numeric value representing a maximum threshold for Ks values.
`low`	A numeric value specifying the lower quantile for bootstrapping. Default is 0.025.
`up`	A numeric value specifying the upper quantile for bootstrapping. Default is 0.975.
`bs`	An integer specifying the number of bootstrap iterations. Default is 1000.

Value

A list containing computed relative rates and their confidence intervals.

`remove_inner_stop_codon_sequence` *Remove Genes Contain Stop Codons within the Sequence*

Description

This function removes the gene contains stop codons (TAA, TAG, TGA, taa, tag, tga) within its sequence.

Usage

```
remove_inner_stop_codon_sequence(sequence)
```

Arguments

`sequence`	A nucleotide sequence as a character string.

Value

A character string or NULL.

`remove_old_dirs` *Remove directories older than a specified day*

Description

This function removes directories in the specified base directory that are older than a specified maximum age in days. It logs the removed directories and any errors encountered during removal.

Usage

```
remove_old_dirs(
  base_dir,
  max_age_in_days = 3,
  log_file = "remove_old_dirs.log",
  verbose = FALSE
)
```

Arguments

`base_dir`	The base directory to search for old directories.
`max_age_in_days`	The maximum age (in days) for directories to be considered old.
`log_file`	The name of the log file to store information about removed directories and errors.
`verbose`	A logical value indicating whether to print messages to the console.

Value

The function does not return anything. It logs information about removed directories and errors.

`replace_informal_name_to_latin_name`
Replace Informal Names with Latin Names

Description

This function takes a data frame `names_df` containing "latin_name" and "informal_name" columns and an `input` string as input. It replaces informal species names in the `input` string with their corresponding Latin names based on the information in `names_df`. If the `input` string contains underscores ("_"), it assumes a comparison between two species and replaces both informal names. Otherwise, it replaces the informal name in the `input` string.

Usage

```
replace_informal_name_to_latin_name(names_df, input)
```

Arguments

`names_df`	A data frame with "latin_name" and "informal_name" columns.
`input`	The input string that may contain informal species names.

Value

A modified input string with informal names replaced by Latin names.

`resampleKsDistribution`
Resample a Ks Distribution

Description

This function resamples a given Ks (synonymous substitution rates) distribution.

Usage

```
resampleKsDistribution(ks, maxK = 5)
```

Arguments

`ks`	A numeric vector representing the Ks distribution to be resampled.
`maxK`	A numeric value indicating the maximum Ks value to consider in the distribution.

Value

A numeric vector containing a resampled Ks distribution.

`runshinyWGD` *The main code to run shinyWGD*

Description

The main function to launch the Shiny application for whole genome duplication analysis. This function initializes the app and opens a Shiny interface that allows users to interactively analyze whole-genome duplication data.

Usage

```
runshinyWGD()
```

Value

No return value. This function is called for side effects, which include starting the Shiny application. The function launches a Shiny app in a web browser, where users can interact with the whole genome duplication analysis.

`run_emmix_kmeas` *A wrapper to run EM analysis of \(ln\) Ks values with k-means*

Description

A wrapper to run EM analysis of \(ln\) Ks values with k-means

Usage

```
run_emmix_kmeas(v, k.centers = 2, k.nstart = 500)
```

Arguments

`v`	A list include a vector of Ks values namely `ks_value`, and a bolean variable namely `log`
`k.centers`	Number of k-means centers, default 2.
`k.nstart`	Number of random start of k-means clustering, default 10. For a formal analysis, it is recommended to use 500.

Value

A list, i.e., the original output of mclust::emV

SignifFeatureRegion *SignifFeatureRegion*

Description

This function computes the significance of features based on gradient and curvature analysis.

Usage

```
SignifFeatureRegion(
  n,
  d,
  gcounts,
  gridsize,
  dest,
  bandwidth,
  signifLevel,
  range.x,
  grad = TRUE,
  curv = TRUE,
  neg.curv.only = TRUE
)
```

Arguments

n	The sample size.
d	The dimensionality of the data.
gcounts	A numeric vector representing data counts.
gridsize	A numeric vector specifying the grid size.
dest	A kernel density estimate.
bandwidth	The bandwidth parameter.
signifLevel	The significance level.
range.x	The range of x values.
grad	A logical value indicating whether to compute the gradient significance.
curv	A logical value indicating whether to compute the curvature significance.
neg.curv.only	A logical value indicating whether to consider negative curvature only.

Value

A list containing the significance results for gradient and curvature.

SiZer *SiZer (Significant Zero Crossings)*

Description

The SiZer (Significant Zero Crossings) method is a technique used for assessing the statistical significance of zero crossings in data density estimation.

Usage

```
SiZer(x, bw, gridsize, signifLevel = 0.05)
```

Arguments

x	A numeric vector containing the data for which you want to calculate SiZer.
bw	Bandwidth parameter for kernel density estimation. If not provided, default values are used.
gridsize	A vector specifying the grid size for SiZer. Default is c(401, 151).
signifLevel	The significance level for SiZer. Default is 0.05.

Value

A list containing SiZer results, including the SiZer curve, the SiZer map, and the bandwidth.

symconv.ks *symconv.ks*

Description

Perform symmetric convolution using FFT.

Usage

```
symconv.ks(rr, ss, skewflag)
```

Arguments

rr	The first input vector.
ss	The second input vector.
skewflag	A scalar value to apply skew correction.

Value

A vector representing the result of the symmetric convolution.

symconv2D.ks *symconv2D.ks*

Description

Perform symmetric 2D convolution using FFT.

Usage

```
symconv2D.ks(rr, ss, skewflag = rep(1, 2))
```

Arguments

rr	The first input matrix.
ss	The second input matrix.
skewflag	A vector of two scalar values for skew correction along each dimension.

Value

A matrix representing the result of the symmetric 2D convolution.

symconv3D.ks *symconv3D.ks*

Description

Perform symmetric 3D convolution using FFT.

Usage

```
symconv3D.ks(rr, ss, skewflag = rep(1, 3))
```

Arguments

rr	The first input 3D array.
ss	The second input 3D array.
skewflag	A vector of three scalar values for skew correction along each dimension.

Value

A 3D array representing the result of the symmetric 3D convolution.

`symconv4D.ks` *symconv4D.ks*

Description

Perform symmetric 4D convolution using FFT.

Usage

```
symconv4D.ks(rr, ss, skewflag = rep(1, 4), fftflag = rep(TRUE, 2))
```

Arguments

`rr`	The first input 4D array.
`ss`	The second input 4D array.
`skewflag`	A vector of four scalar values for skew correction along each dimension.
`fftflag`	A vector of two Boolean values for FFT flag.

Value

A 4D array representing the result of the symmetric 4D convolution.

`TimeTreeFecher` *Extracts a timetree from TimeTree.org based on species names.*

Description

This function takes a file with species names as input and a prefix to define the output.

Usage

```
TimeTreeFecher(input_file, prefix)
```

Arguments

`input_file`	A character string specifying the path to the file containing species names.
`prefix`	A character string providing the prefix for the output file.

Value

A timetree object representing the estimated divergence times between species.

Index