Package ‘GaMaBioMD’

January 12, 2024

Type Package

Title Diversity Analysis for Sequence Data

Version 0.2.0

Author Sandip Garai [aut, cre],
Shrikant Mantri [aut]

Maintainer Sandip Garai <sandipnicksandy@gmail.com>

Description The full form is Garai and Mantri Biological Material Diversity. It is an R package designed for the calculation of biological diversity using sequence data. It simplifies the process by requiring only sample IDs and accession numbers. Whether you're analyzing genetic or microbial diversity, It provides efficient tools for diversity analysis. Serially one should go for the functions as presented here expand_accession_ranges(), get_sequence_information(), preprocess_for_alignment(), write_fasta(), SampleID_vs_NumSequences(), data_sampling(), alignment_info(), compute_average_similarity_matrix(), generate_heatmaps(), clustering_average_similarity(), clustering_percent_similarity(), bubble_plot_count(), bubble_plot_percentage(), tree_average_similarity(), tree_percent_similarity(). Till date there are total 15 functions. More details can be found in Faith (1992) <doi:10.1016/0006-3207(92)91201-3>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.3

Imports traits, dplyr, magrittr, ggplot2, Biostrings, heatmaply,
reshape2, dendextend, stats, ape

NeedsCompilation no

Repository CRAN

Date/Publication 2024-01-12 14:40:02 UTC

R topics documented:

`alignment_info` *Sequence Alignment and Analysis*

Description

Sequence Alignment and Analysis

Usage

```
alignment_info(data, type = "global", verbose = 1)
```

Arguments

`data`	A data frame with 'SequenceID' and 'Sequence' columns.
`type`	A character string specifying the type of sequence alignment ('global', 'local', etc.).
`verbose`	An integer indicating the level of verbosity (0, 1, or 2).

Value

A list containing matrices and results from sequence alignment.

References

Faith, D. P. (1992). Conservation evaluation and phylogenetic diversity. Biological conservation, 61(1), 1-10.

Examples

```
accession_ranges <- list(
  SRU1 = "AJ240966 to AJ240970",
  STU2 = "AB015240 to AB015245",
  WPU13 = "L11934 to L11939",
  INU20 = c("AF277467 to AF277470", "AF333080 to AF333085")
)

# Use the function to expand accession ranges
sam_acc <- expand_accession_ranges(accession_ranges)
print(sam_acc)

# 2 get_sequence_information
accessions_to_query <- sam_acc$accession
seq_info <- get_sequence_information(accessions_to_query, remove_dot_1 = TRUE)
print(seq_info)
result <- preprocess_for_alignment(sam_acc, seq_info)

# Access the resulting data frames
merged_data <- result$merged_data
main_data <- result$main_data
final_data <- result$final_data

# If you want to sample 10% from each SampleID group:
sampled_data <- data_sampling(final_data, sample_proportion = 0.1)

alignment_results <- alignment_info(final_data, type = "global", verbose = 1)

# Access the resulting data frames
score_matrix <- alignment_results$score_matrix
normalized_score_matrix <- alignment_results$normalized_score_matrix

total_aligned_positions_matrix <- alignment_results$total_aligned_positions_matrix
number_of_matching_positions_matrix <- alignment_results$number_of_matching_positions_matrix

percent_similarity_matrix <- alignment_results$percent_similarity_matrix

alignment_results_list <- alignment_results$alignment_results_list

alignment_info_matrix <- alignment_results$alignment_info_matrix

output_directory <- tempdir()

# Save the list of alignment results to an RDS file
saveRDS(alignment_results_list, file.path(output_directory, "alignment_results_list.rds"))

# Save matrices to files
write.table(score_matrix, file.path(output_directory, "score_matrix.txt"), sep = "\t")
```

bubble_plot_count *Bubble Plot Count Function*

Description

This function generates a bubble plot using ggplot2.

Usage

```
bubble_plot_count(
  clustered_data,
  title,
  x_label,
  y_label,
  size_label,
  color_label
)
```

Arguments

clustered_data	A data frame containing clustered data.
title	The title of the plot.
x_label	The label for the x-axis.
y_label	The label for the y-axis.
size_label	The label for the size variable.
color_label	The label for the color variable.

Value

A ggplot object representing the bubble plot.

Examples

```
accession_ranges <- list(
  SRU1 = "AJ240966 to AJ240970",
  STU2 = "AB015240 to AB015245",
  WPU13 = "L11934 to L11939",
  INU20 = c("AF277467 to AF277470", "AF333080 to AF333085")
)

# Use the function to expand accession ranges
sam_acc <- expand_accession_ranges(accession_ranges)
print(sam_acc)

# 2 get_sequence_information
accessions_to_query <- sam_acc$accession
```

```
seq_info <- get_sequence_information(accessions_to_query, remove_dot_1 = TRUE)
print(seq_info)
result <- preprocess_for_alignment(sam_acc, seq_info)

# Access the resulting data frames
merged_data <- result$merged_data
main_data <- result$main_data
final_data <- result$final_data

# If you want to sample 10% from each SampleID group:
sampled_data <- data_sampling(final_data, sample_proportion = 0.1)

alignment_results <- alignment_info(final_data, type = "global", verbose = 1)

# Access the resulting data frames
score_matrix <- alignment_results$score_matrix
normalized_score_matrix <- alignment_results$normalized_score_matrix

total_aligned_positions_matrix <- alignment_results$total_aligned_positions_matrix
number_of_matching_positions_matrix <- alignment_results$number_of_matching_positions_matrix

percent_similarity_matrix <- alignment_results$percent_similarity_matrix

alignment_results_list <- alignment_results$alignment_results_list

alignment_info_matrix <- alignment_results$alignment_info_matrix

output_directory <- tempdir()

# Save the list of alignment results to an RDS file
saveRDS(alignment_results_list, file.path(output_directory, "alignment_results_list.rds"))

# Save matrices to files
write.table(score_matrix, file.path(output_directory, "score_matrix.txt"), sep = "\t")
average_percent_similarity <- compute_average_similarity_matrix(percent_similarity_matrix)
print(average_percent_similarity)

output_directory <- tempdir()
width_inch <- 8
height_inch <- 6
dpi <- 300

clustering_result <- clustering_average_similarity(average_percent_similarity)

# Extract the dendrogram and clustered data
dend_colored <- clustering_result$dendrogram
clustered_data <- clustering_result$clustered_data
Cluster_SampleID_Percentage <- clustering_result$Cluster_SampleID_Percentage
Cluster_TotalPercentage <- clustering_result$Cluster_TotalPercentage

tiff_file <- file.path(output_directory, "6. hierarchical_clustering_dendrogram_colored.tiff")

# Save the dendrogram as a TIFF image
```

```
tiff(tiff_file, width = width_inch, height = height_inch, units = "in", res = dpi)
plot(dend_colored, main = "Colored Hierarchical Clustering Dendrogram")
dev.off()

# Save the clustered data frame to a CSV file
write.csv(clustered_data, file.path(output_directory, "7. clustered_data.csv"), row.names = FALSE)

# Example usage with clustered_data
clustered_data <- clustered_data # Load or generate your clustered data
bubble_plot_count <- bubble_plot_count(clustered_data = clustered_data,
                                       title = "Bubble Plot of Clusters",
                                       x_label = "Clusters",
                                       y_label = "Sample ID",
                                       size_label = "Count",
                                       color_label = "Sample ID")

# Save the bubble plot as a TIFF image
output_directory <- tempdir()
width_inch <- 8
height_inch <- 6
dpi <- 300

# Set the file name for the TIFF image
tiff_file <- file.path(output_directory, "bubble_plot_count.tiff")

# Open the TIFF device
tiff(tiff_file, width = width_inch, height = height_inch, units = "in", res = dpi)

# Print and save the bubble plot
print(bubble_plot_count)

# Close the TIFF device
dev.off()
```

bubble_plot_percentage — *Bubble Plot Percentage Function*

Description

This function generates a bubble plot using ggplot2 based on percentage data.

Usage

```
bubble_plot_percentage(
  Cluster_SampleID_Percentage,
  title,
  x_label,
  y_label,
  size_label,
  color_label
)
```

Arguments

`Cluster_SampleID_Percentage`	A data frame containing cluster, sample ID, and percentage data.
`title`	The title of the plot.
`x_label`	The label for the x-axis.
`y_label`	The label for the y-axis.
`size_label`	The label for the size variable.
`color_label`	The label for the color variable.

Value

A ggplot object representing the bubble plot.

Examples

```
accession_ranges <- list(
  SRU1 = "AJ240966 to AJ240970",
  STU2 = "AB015240 to AB015245",
  WPU13 = "L11934 to L11939",
  INU20 = c("AF277467 to AF277470", "AF333080 to AF333085")
)

# Use the function to expand accession ranges
sam_acc <- expand_accession_ranges(accession_ranges)
print(sam_acc)

# 2 get_sequence_information
accessions_to_query <- sam_acc$accession
seq_info <- get_sequence_information(accessions_to_query, remove_dot_1 = TRUE)
print(seq_info)
result <- preprocess_for_alignment(sam_acc, seq_info)

# Access the resulting data frames
merged_data <- result$merged_data
main_data <- result$main_data
final_data <- result$final_data

# If you want to sample 10% from each SampleID group:
sampled_data <- data_sampling(final_data, sample_proportion = 0.1)

alignment_results <- alignment_info(final_data, type = "global", verbose = 1)

# Access the resulting data frames
score_matrix <- alignment_results$score_matrix
```

```
normalized_score_matrix <- alignment_results$normalized_score_matrix

total_aligned_positions_matrix <- alignment_results$total_aligned_positions_matrix
number_of_matching_positions_matrix <- alignment_results$number_of_matching_positions_matrix

percent_similarity_matrix <- alignment_results$percent_similarity_matrix

alignment_results_list <- alignment_results$alignment_results_list

alignment_info_matrix <- alignment_results$alignment_info_matrix

output_directory <- tempdir()

# Save the list of alignment results to an RDS file
saveRDS(alignment_results_list, file.path(output_directory, "alignment_results_list.rds"))

# Save matrices to files
write.table(score_matrix, file.path(output_directory, "score_matrix.txt"), sep = "\t")
average_percent_similarity <- compute_average_similarity_matrix(percent_similarity_matrix)
print(average_percent_similarity)

output_directory <- tempdir()
width_inch <- 8
height_inch <- 6
dpi <- 300

clustering_result <- clustering_percent_similarity(percent_similarity_matrix)

# Extract the dendrogram and clustered data
dend_colored <- clustering_result$dendrogram
clustered_data <- clustering_result$clustered_data
Cluster_SampleID_Percentage <- clustering_result$Cluster_SampleID_Percentage
Cluster_TotalPercentage <- clustering_result$Cluster_TotalPercentage

tiff_file <- file.path(output_directory, "6. hierarchical_clustering_dendrogram_colored.tiff")

# Save the dendrogram as a TIFF image
tiff(tiff_file, width = width_inch, height = height_inch, units = "in", res = dpi)
plot(dend_colored, main = "Colored Hierarchical Clustering Dendrogram")
dev.off()

# Save the clustered data frame to a CSV file
write.csv(clustered_data, file.path(output_directory, "7. clustered_data.csv"), row.names = FALSE)

# Example usage with Cluster_SampleID_Percentage
Cluster_SampleID_Percentage <- Cluster_SampleID_Percentage
bubble_plot_percentage <- bubble_plot_percentage(Cluster_SampleID_Percentage,
                                                title = "Bubble Plot",
                                                x_label = "Clusters",
                                                y_label = "Sample ID",
                                                size_label = "Percentage",
                                                color_label = "Sample ID")
```

```
# Save the bubble plot as a TIFF image
output_directory <- tempdir()
width_inch <- 8
height_inch <- 6
dpi <- 300

# Set the file name for the TIFF image
tiff_file <- file.path(output_directory, "bubble_plot_percentage.tiff")

# Open the TIFF device
tiff(tiff_file, width = width_inch, height = height_inch, units = "in", res = dpi)

# Print and save the bubble plot
print(bubble_plot_percentage)

# Close the TIFF device
dev.off()
```

`clustering_average_similarity` *Perform Hierarchical Clustering for average similarity matrix*

Description

This function performs hierarchical clustering based on a similarity matrix and provides additional information such as a colored dendrogram, clustered data, sample percentage within clusters, and total percentage for each cluster.

Usage

```
clustering_average_similarity(
  similarity_matrix,
  num_clusters_option = 10,
  cut_height_option = 0.2
)
```

Arguments

`similarity_matrix`	A matrix containing similarity values between SampleIDs.
`num_clusters_option`	The desired number of clusters. Default is 10.
`cut_height_option`	The height at which the dendrogram should be cut to obtain clusters. Default is 0.2.

Value

A list containing the dendrogram, clustered data, sample percentage within clusters, and total percentage for each cluster.

Examples

```
accession_ranges <- list(
  SRU1 = "AJ240966 to AJ240970",
  STU2 = "AB015240 to AB015245",
  WPU13 = "L11934 to L11939",
  INU20 = c("AF277467 to AF277470", "AF333080 to AF333085")
)

# Use the function to expand accession ranges
sam_acc <- expand_accession_ranges(accession_ranges)
print(sam_acc)

# 2 get_sequence_information
accessions_to_query <- sam_acc$accession
seq_info <- get_sequence_information(accessions_to_query, remove_dot_1 = TRUE)
print(seq_info)
result <- preprocess_for_alignment(sam_acc, seq_info)

# Access the resulting data frames
merged_data <- result$merged_data
main_data <- result$main_data
final_data <- result$final_data

# If you want to sample 10% from each SampleID group:
sampled_data <- data_sampling(final_data, sample_proportion = 0.1)

alignment_results <- alignment_info(final_data, type = "global", verbose = 1)

# Access the resulting data frames
score_matrix <- alignment_results$score_matrix
normalized_score_matrix <- alignment_results$normalized_score_matrix

total_aligned_positions_matrix <- alignment_results$total_aligned_positions_matrix
number_of_matching_positions_matrix <- alignment_results$number_of_matching_positions_matrix

percent_similarity_matrix <- alignment_results$percent_similarity_matrix

alignment_results_list <- alignment_results$alignment_results_list

alignment_info_matrix <- alignment_results$alignment_info_matrix

output_directory <- tempdir()

# Save the list of alignment results to an RDS file
saveRDS(alignment_results_list, file.path(output_directory, "alignment_results_list.rds"))
```

```
# Save matrices to files
write.table(score_matrix, file.path(output_directory, "score_matrix.txt"), sep = "\t")
average_percent_similarity <- compute_average_similarity_matrix(percent_similarity_matrix)
print(average_percent_similarity)

output_directory <- tempdir()
width_inch <- 8
height_inch <- 6
dpi <- 300

clustering_result <- clustering_average_similarity(average_percent_similarity)

# Extract the dendrogram and clustered data
dend_colored <- clustering_result$dendrogram
clustered_data <- clustering_result$clustered_data
Cluster_SampleID_Percentage <- clustering_result$Cluster_SampleID_Percentage
Cluster_TotalPercentage <- clustering_result$Cluster_TotalPercentage

tiff_file <- file.path(output_directory, "6. hierarchical_clustering_dendrogram_colored.tiff")

# Save the dendrogram as a TIFF image
tiff(tiff_file, width = width_inch, height = height_inch, units = "in", res = dpi)
plot(dend_colored, main = "Colored Hierarchical Clustering Dendrogram")
dev.off()

# Save the clustered data frame to a CSV file
write.csv(clustered_data, file.path(output_directory, "7. clustered_data.csv"), row.names = FALSE)
```

clustering_percent_similarity
Perform Hierarchical Clustering for Percent Similarity

Description

This function performs hierarchical clustering based on a percent similarity matrix and provides additional information such as a colored dendrogram, clustered data, sample percentage within clusters, and total percentage for each cluster.

Usage

```
clustering_percent_similarity(
  similarity_matrix,
  num_clusters_option = 10,
  cut_height_option = 0.2,
  add_to_final_data = TRUE
)
```

Arguments

`similarity_matrix`	A matrix containing percent similarity values between SampleIDs.
`num_clusters_option`	The desired number of clusters. Default is 10.
`cut_height_option`	The height at which the dendrogram should be cut to obtain clusters. Default is 0.2.
`add_to_final_data`	A logical value indicating whether to add the clustering results to the final data. Default is TRUE.

Value

A list containing the dendrogram, clustered data, sample percentage within clusters, and total percentage for each cluster.

Examples

```
accession_ranges <- list(
  SRU1 = "AJ240966 to AJ240970",
  STU2 = "AB015240 to AB015245",
  WPU13 = "L11934 to L11939",
  INU20 = c("AF277467 to AF277470", "AF333080 to AF333085")
)

# Use the function to expand accession ranges
sam_acc <- expand_accession_ranges(accession_ranges)
print(sam_acc)

# 2 get_sequence_information
accessions_to_query <- sam_acc$accession
seq_info <- get_sequence_information(accessions_to_query, remove_dot_1 = TRUE)
print(seq_info)
result <- preprocess_for_alignment(sam_acc, seq_info)

# Access the resulting data frames
merged_data <- result$merged_data
main_data <- result$main_data
final_data <- result$final_data

# If you want to sample 10% from each SampleID group:
sampled_data <- data_sampling(final_data, sample_proportion = 0.1)

alignment_results <- alignment_info(final_data, type = "global", verbose = 1)

# Access the resulting data frames
score_matrix <- alignment_results$score_matrix
normalized_score_matrix <- alignment_results$normalized_score_matrix
```

```
total_aligned_positions_matrix <- alignment_results$total_aligned_positions_matrix
number_of_matching_positions_matrix <- alignment_results$number_of_matching_positions_matrix

percent_similarity_matrix <- alignment_results$percent_similarity_matrix

alignment_results_list <- alignment_results$alignment_results_list

alignment_info_matrix <- alignment_results$alignment_info_matrix

output_directory <- tempdir()

# Save the list of alignment results to an RDS file
saveRDS(alignment_results_list, file.path(output_directory, "alignment_results_list.rds"))

# Save matrices to files
write.table(score_matrix, file.path(output_directory, "score_matrix.txt"), sep = "\t")
average_percent_similarity <- compute_average_similarity_matrix(percent_similarity_matrix)
print(average_percent_similarity)

output_directory <- tempdir()
width_inch <- 8
height_inch <- 6
dpi <- 300

clustering_result <- clustering_percent_similarity(percent_similarity_matrix)

# Extract the dendrogram and clustered data
dend_colored <- clustering_result$dendrogram
clustered_data <- clustering_result$clustered_data
Cluster_SampleID_Percentage <- clustering_result$Cluster_SampleID_Percentage
Cluster_TotalPercentage <- clustering_result$Cluster_TotalPercentage

tiff_file <- file.path(output_directory, "6. hierarchical_clustering_dendrogram_colored.tiff")

# Save the dendrogram as a TIFF image
tiff(tiff_file, width = width_inch, height = height_inch, units = "in", res = dpi)
plot(dend_colored, main = "Colored Hierarchical Clustering Dendrogram")
dev.off()

# Save the clustered data frame to a CSV file
write.csv(clustered_data, file.path(output_directory, "7. clustered_data.csv"), row.names = FALSE)
```

compute_average_similarity_matrix

Compute Average Similarity Matrix

Description

This function computes the average similarity matrix based on the percent similarity matrix.

Usage

```
compute_average_similarity_matrix(percent_similarity_matrix)
```

Arguments

`percent_similarity_matrix`	A matrix containing percent similarity values between sequences. Rows and columns should be labeled with SequenceIDs.

Value

A matrix containing average similarity values between SampleIDs.

Examples

```
accession_ranges <- list(
  SRU1 = "AJ240966 to AJ240970",
  STU2 = "AB015240 to AB015245",
  WPU13 = "L11934 to L11939",
  INU20 = c("AF277467 to AF277470", "AF333080 to AF333085")
)

# Use the function to expand accession ranges
sam_acc <- expand_accession_ranges(accession_ranges)
print(sam_acc)

# 2 get_sequence_information
accessions_to_query <- sam_acc$accession
seq_info <- get_sequence_information(accessions_to_query, remove_dot_1 = TRUE)
print(seq_info)
result <- preprocess_for_alignment(sam_acc, seq_info)

# Access the resulting data frames
merged_data <- result$merged_data
main_data <- result$main_data
final_data <- result$final_data

# If you want to sample 10% from each SampleID group:
sampled_data <- data_sampling(final_data, sample_proportion = 0.1)

alignment_results <- alignment_info(final_data, type = "global", verbose = 1)

# Access the resulting data frames
score_matrix <- alignment_results$score_matrix
normalized_score_matrix <- alignment_results$normalized_score_matrix

total_aligned_positions_matrix <- alignment_results$total_aligned_positions_matrix
number_of_matching_positions_matrix <- alignment_results$number_of_matching_positions_matrix

percent_similarity_matrix <- alignment_results$percent_similarity_matrix
```

```
alignment_results_list <- alignment_results$alignment_results_list

alignment_info_matrix <- alignment_results$alignment_info_matrix

output_directory <- tempdir()

# Save the list of alignment results to an RDS file
saveRDS(alignment_results_list, file.path(output_directory, "alignment_results_list.rds"))

# Save matrices to files
write.table(score_matrix, file.path(output_directory, "score_matrix.txt"), sep = "\t")
average_percent_similarity <- compute_average_similarity_matrix(percent_similarity_matrix)
print(average_percent_similarity)
```

`data_sampling` *Samples data from each SampleID group if specified, otherwise uses the final data.*

Description

This function takes a data frame with 'SampleID' and 'SequenceID' columns and either returns the original data frame (if sample_proportion is NULL) or samples a specified proportion from each SampleID group.

Usage

```
data_sampling(final_data, sample_proportion = NULL)
```

Arguments

`final_data`	A data frame with 'SampleID' and 'SequenceID' columns.
`sample_proportion`	Proportion of data to sample from each SampleID group. If NULL, the original data frame is returned.

Value

A data frame either with the original data or sampled data based on the specified proportion.

Examples

```
accession_ranges <- list(
  SRU1 = "AJ240966 to AJ240970",
  STU2 = "AB015240 to AB015245",
  WPU13 = "L11934 to L11939",
  INU20 = c("AF277467 to AF277470", "AF333080 to AF333085")
)
```

```
# Use the function to expand accession ranges
sam_acc <- expand_accession_ranges(accession_ranges)
print(sam_acc)

# 2 get_sequence_information
accessions_to_query <- sam_acc$accession
seq_info <- get_sequence_information(accessions_to_query, remove_dot_1 = TRUE)
print(seq_info)
result <- preprocess_for_alignment(sam_acc, seq_info)

# Access the resulting data frames
merged_data <- result$merged_data
main_data <- result$main_data
final_data <- result$final_data # use final_data

# If you want to sample 10% from each SampleID group:
sampled_data <- data_sampling(final_data, sample_proportion = 0.1)
```

`expand_accession_ranges` *This function expands accession number ranges, creating a data frame with sample and accession numbers.*

Description

This function expands accession number ranges, creating a data frame with sample and accession numbers.

Usage

```
expand_accession_ranges(accession_ranges)
```

Arguments

`accession_ranges`	A named list where each element represents an accession range. The names of the list elements should correspond to sample names.

Value

A data frame with columns ’sample’ and ’accession’.

Examples

```
# Example of defining accession number ranges.
accession_ranges <- list(
  SRU1 = "AJ240966 to AJ240970",
  STU2 = "AB015240 to AB015250",
  WRU8 = c("AF245628", "AF353208 to AF353210"),
  WPU13 = "L11934 to L11940",
  INU20 = c("AF277467 to AF277470", "AF333080 to AF333086")
)

# Use the function to expand accession ranges
sam_acc <- expand_accession_ranges(accession_ranges)
print(sam_acc)
```

`generate_heatmaps` *Generate Heatmaps*

Description

This function generates interactive and static heatmaps based on the average similarity matrix.

Usage

```
generate_heatmaps(
  average_percent_similarity,
  html_title = "Heatmap of Average Similarity",
  tiff_title = "Heatmap of Average Similarity",
  xlab = "SampleID",
  ylab = "SampleID",
  width_inch = 8,
  height_inch = 6,
  dpi = 300
)
```

Arguments

`average_percent_similarity`	A matrix containing average similarity values between SampleIDs. Rows and columns should be labeled with SampleIDs.
`html_title`	Title for the interactive heatmap (HTML version).
`tiff_title`	Title for the static heatmap (TIFF version).
`xlab`	Label for the x-axis.
`ylab`	Label for the y-axis.
`width_inch`	Width of the heatmap in inches.
`height_inch`	Height of the heatmap in inches.
`dpi`	Dots per inch for the static heatmap.

Value

A list containing the file names of the generated heatmaps.

Examples

```
accession_ranges <- list(
  SRU1 = "AJ240966 to AJ240970",
  STU2 = "AB015240 to AB015245",
  WPU13 = "L11934 to L11939",
  INU20 = c("AF277467 to AF277470", "AF333080 to AF333085")
)

# Use the function to expand accession ranges
sam_acc <- expand_accession_ranges(accession_ranges)
print(sam_acc)

# 2 get_sequence_information
accessions_to_query <- sam_acc$accession
seq_info <- get_sequence_information(accessions_to_query, remove_dot_1 = TRUE)
print(seq_info)
result <- preprocess_for_alignment(sam_acc, seq_info)

# Access the resulting data frames
merged_data <- result$merged_data
main_data <- result$main_data
final_data <- result$final_data

# If you want to sample 10% from each SampleID group:
sampled_data <- data_sampling(final_data, sample_proportion = 0.1)

alignment_results <- alignment_info(final_data, type = "global", verbose = 1)

# Access the resulting data frames
score_matrix <- alignment_results$score_matrix
normalized_score_matrix <- alignment_results$normalized_score_matrix

total_aligned_positions_matrix <- alignment_results$total_aligned_positions_matrix
number_of_matching_positions_matrix <- alignment_results$number_of_matching_positions_matrix

percent_similarity_matrix <- alignment_results$percent_similarity_matrix

alignment_results_list <- alignment_results$alignment_results_list

alignment_info_matrix <- alignment_results$alignment_info_matrix

output_directory <- tempdir()

# Save the list of alignment results to an RDS file
saveRDS(alignment_results_list, file.path(output_directory, "alignment_results_list.rds"))

# Save matrices to files
write.table(score_matrix, file.path(output_directory, "score_matrix.txt"), sep = "\t")
average_percent_similarity <- compute_average_similarity_matrix(percent_similarity_matrix)
print(average_percent_similarity)
```

```
output_directory <- tempdir()
width_inch <- 8
height_inch <- 6
dpi <- 300

heatmap_files <- generate_heatmaps(average_percent_similarity)

# Save the interactive heatmap as an HTML file ####
html <- file.path(output_directory, "5. heatmap.html")
# htmlwidgets should be installed and loaded
# htmlwidgets::saveWidget(heatmap_files$html, file = html)

# save the TIFFE images ####

# heatmap_tiff_file1
tiff1 <- file.path(output_directory, "5. heatmap1.tiff")
tiff(tiff1, width = width_inch, height = height_inch, units = "in", res = dpi)
heatmap(as.matrix(average_percent_similarity), main = "Heatmap of Average Similarity")

# Close the TIFF device
dev.off()

# heatmap_tiff_file2
tiff2 <- file.path(output_directory, "5. heatmap2.tiff")

# Open the TIFF device
tiff(tiff2, width = width_inch, height = height_inch, units = "in", res = dpi)
print(heatmap_files$tiff2)

# Close the TIFF device
dev.off()

# heatmap_tiff_file3
# tiff3 <- file.path(output_directory, "5. heatmap3.tiff")

# Open the TIFF device and create the heatmap.2 with hierarchical clustering dendrogram
# gplots should be installed and loaded
# gplots::heatmap.2(as.matrix(average_percent_similarity),
#          dendrogram = "row",
#          Colv = "Rowv",
#          scale = "row",
#          main = "Heatmap of Average Similarity")

# Close the TIFF device
# dev.off()
```

get_sequence_information

Retrieves sequence information for a given list of accessions.

Description

Retrieves sequence information for a given list of accessions.

Usage

```
get_sequence_information(accession, remove_dot_1 = FALSE)
```

Arguments

`accession`	accession number.
`remove_dot_1`	Logical. If TRUE, removes '.1' from accession numbers.

Value

A data frame containing sequence information for the given accessions.

Examples

```
accession_ranges <- list(
  SRU1 = "AJ240966 to AJ240970",
  STU2 = "AB015240 to AB015245",
  WPU13 = "L11934 to L11939",
  INU20 = c("AF277467 to AF277470", "AF333080 to AF333085")
)

# Use the function to expand accession ranges
sam_acc <- expand_accession_ranges(accession_ranges)
print(sam_acc)

# 2 get_sequence_information
accessions_to_query <- sam_acc$accession
seq_info <- get_sequence_information(accessions_to_query, remove_dot_1 = TRUE)
print(seq_info)
```

`preprocess_for_alignment` *Preprocesses data for sequence alignment.*

Description

This function merges sample and accession information with sequence information, filters out rows with missing sequences, and extracts relevant columns for the final data.

Usage

```
preprocess_for_alignment(sam_acc, seq_info)
```

Arguments

`sam_acc`	A data frame containing sample and accession information.
`seq_info`	A data frame containing sequence information.

Value

A list containing the resulting data frames: 'merged_data', 'main_data', 'final_data'.

Examples

```
accession_ranges <- list(
  SRU1 = "AJ240966 to AJ240970",
  STU2 = "AB015240 to AB015245",
  WPU13 = "L11934 to L11939",
  INU20 = c("AF277467 to AF277470", "AF333080 to AF333085")
)

# Use the function to expand accession ranges
sam_acc <- expand_accession_ranges(accession_ranges)
print(sam_acc)

# 2 get_sequence_information
accessions_to_query <- sam_acc$accession
seq_info <- get_sequence_information(accessions_to_query, remove_dot_1 = TRUE)
print(seq_info)
result <- preprocess_for_alignment(sam_acc, seq_info)

# Access the resulting data frames
merged_data <- result$merged_data
main_data <- result$main_data
final_data <- result$final_data
```

`SampleID_vs_NumSequences` *Plots the number of sequences and the probability of sequences per SampleID.*

Description

This function takes a data frame with 'SampleID' and 'SequenceID' columns and creates two bar plots. The first plot shows the number of sequences per SampleID, and the second plot shows the probability of sequences per SampleID.

Usage

```
SampleID_vs_NumSequences(final_data)
```

Arguments

`final_data`	A data frame with 'SampleID' and 'SequenceID' columns.

Value

A list containing two ggplot2 plots: 'plot_num_sequences' and 'plot_prob'.

Examples

```
accession_ranges <- list(
  SRU1 = "AJ240966 to AJ240970",
  STU2 = "AB015240 to AB015245",
  WPU13 = "L11934 to L11939",
  INU20 = c("AF277467 to AF277470", "AF333080 to AF333085")
)

# Use the function to expand accession ranges
sam_acc <- expand_accession_ranges(accession_ranges)
print(sam_acc)

# 2 get_sequence_information
accessions_to_query <- sam_acc$accession
seq_info <- get_sequence_information(accessions_to_query, remove_dot_1 = TRUE)
print(seq_info)
result <- preprocess_for_alignment(sam_acc, seq_info)

# Access the resulting data frames
merged_data <- result$merged_data
main_data <- result$main_data
final_data <- result$final_data

# Example usage
plots <- SampleID_vs_NumSequences(final_data)

output_directory <- tempdir()
# Set the file name for the TIFF images
tiff_file_num_sequences <- file.path(output_directory, "0. SampleID_vs_NumSequences.tiff")
tiff_file_prob <- file.path(output_directory, "0. SampleID_vs_Probability.tiff")

# Set the width, height, and DPI parameters
width_inch <- 8
height_inch <- 8
dpi <- 300

# Open the TIFF devices and save the plots
tiff(tiff_file_num_sequences, width = width_inch, height = height_inch, units = "in", res = dpi)
print(plots$plot_num_sequences)
dev.off()

tiff(tiff_file_prob, width = width_inch, height = height_inch, units = "in", res = dpi)
print(plots$plot_prob)
```

```
dev.off()
```

`tree_average_similarity` *Generate Phylogenetic Tree and Color Palette for Average Similarity*

Description

This function generates a Neighbor-Joining phylogenetic tree and a color palette based on the average similarity matrix.

Usage

```
tree_average_similarity(similarity_matrix)
```

Arguments

`similarity_matrix`	A matrix containing pairwise similarities between samples.

Value

A list containing the phylogenetic tree and a color palette.

Examples

```
accession_ranges <- list(
  SRU1 = "AJ240966 to AJ240970",
  STU2 = "AB015240 to AB015245",
  WPU13 = "L11934 to L11939",
  INU20 = c("AF277467 to AF277470", "AF333080 to AF333085")
)

# Use the function to expand accession ranges
sam_acc <- expand_accession_ranges(accession_ranges)
print(sam_acc)

# 2 get_sequence_information
accessions_to_query <- sam_acc$accession
seq_info <- get_sequence_information(accessions_to_query, remove_dot_1 = TRUE)
print(seq_info)
result <- preprocess_for_alignment(sam_acc, seq_info)

# Access the resulting data frames
merged_data <- result$merged_data
main_data <- result$main_data
final_data <- result$final_data
```

```
# If you want to sample 10% from each SampleID group:
sampled_data <- data_sampling(final_data, sample_proportion = 0.1)

alignment_results <- alignment_info(final_data, type = "global", verbose = 1)

# Access the resulting data frames
score_matrix <- alignment_results$score_matrix
normalized_score_matrix <- alignment_results$normalized_score_matrix

total_aligned_positions_matrix <- alignment_results$total_aligned_positions_matrix
number_of_matching_positions_matrix <- alignment_results$number_of_matching_positions_matrix

percent_similarity_matrix <- alignment_results$percent_similarity_matrix

alignment_results_list <- alignment_results$alignment_results_list

alignment_info_matrix <- alignment_results$alignment_info_matrix

output_directory <- tempdir()

# Save the list of alignment results to an RDS file
saveRDS(alignment_results_list, file.path(output_directory, "alignment_results_list.rds"))

# Save matrices to files
write.table(score_matrix, file.path(output_directory, "score_matrix.txt"), sep = "\t")
average_percent_similarity <- compute_average_similarity_matrix(percent_similarity_matrix)
print(average_percent_similarity)

output_directory <- tempdir()
width_inch <- 8
height_inch <- 6
dpi <- 300

clustering_result <- clustering_average_similarity(average_percent_similarity)

# Extract the dendrogram and clustered data
dend_colored <- clustering_result$dendrogram
clustered_data <- clustering_result$clustered_data
Cluster_SampleID_Percentage <- clustering_result$Cluster_SampleID_Percentage
Cluster_TotalPercentage <- clustering_result$Cluster_TotalPercentage

tiff_file <- file.path(output_directory, "6. hierarchical_clustering_dendrogram_colored.tiff")

# Save the dendrogram as a TIFF image
tiff(tiff_file, width = width_inch, height = height_inch, units = "in", res = dpi)
plot(dend_colored, main = "Colored Hierarchical Clustering Dendrogram")
dev.off()

# Save the clustered data frame to a CSV file
write.csv(clustered_data, file.path(output_directory, "7. clustered_data.csv"), row.names = FALSE)

# Example usage with similarity_matrix
result <- tree_average_similarity(average_percent_similarity)
```

```
tree <- result$tree
color_palette <- result$color_palette

output_directory <- tempdir()

tree_newick <- file.path(output_directory, "phylogenetic_tree_nj.nwk")

# Save the phylogenetic tree as a Newick file
# ape::write.tree(tree, file = tree_newick)

# Save the phylogenetic tree as a TIFF image
width_inch <- 8
height_inch <- 6
dpi <- 300

# Set the file name for the TIFF image
tiff_file <- file.path(output_directory, "phylogenetic_tree.tiff")

# Open the TIFF device
tiff(tiff_file, width = width_inch, height = height_inch, units = "in", res = dpi)

# Plot the phylogenetic tree vertically
plot(tree, main = "Neighbor-Joining Tree", cex = 1, direction = "downward")

# Close the TIFF device
dev.off()

# Set the file name for the TIFF image with rainbow-colored branches
tiff_file_rainbow <- file.path(output_directory, "phylogenetic_tree_rainbow.tiff")

# Open the TIFF device
tiff(tiff_file_rainbow, width = width_inch, height = height_inch, units = "in", res = dpi)

# Plot the phylogenetic tree vertically with rainbow-colored branches
plot(tree, main = "NJ Tree", cex = 1, direction = "downward", tip.color = color_palette)

# Close the TIFF device
dev.off()
```

---

tree_percent_similarity

Generate Phylogenetic Tree and Color Palette for Percent Similarity Matrix

---

Description

This function generates a Neighbor-Joining phylogenetic tree and a color palette based on the percent similarity matrix.

Usage

```
tree_percent_similarity(percent_similarity_matrix)
```

Arguments

`percent_similarity_matrix`	A matrix containing pairwise percent similarities between samples.

Value

A list containing the phylogenetic tree and a color palette.

Examples

```
accession_ranges <- list(
  SRU1 = "AJ240966 to AJ240970",
  STU2 = "AB015240 to AB015245",
  WPU13 = "L11934 to L11939",
  INU20 = c("AF277467 to AF277470", "AF333080 to AF333085")
)

# Use the function to expand accession ranges
sam_acc <- expand_accession_ranges(accession_ranges)
print(sam_acc)

# 2 get_sequence_information
accessions_to_query <- sam_acc$accession
seq_info <- get_sequence_information(accessions_to_query, remove_dot_1 = TRUE)
print(seq_info)
result <- preprocess_for_alignment(sam_acc, seq_info)

# Access the resulting data frames
merged_data <- result$merged_data
main_data <- result$main_data
final_data <- result$final_data

# If you want to sample 10% from each SampleID group:
sampled_data <- data_sampling(final_data, sample_proportion = 0.1)

alignment_results <- alignment_info(final_data, type = "global", verbose = 1)

# Access the resulting data frames
score_matrix <- alignment_results$score_matrix
normalized_score_matrix <- alignment_results$normalized_score_matrix

total_aligned_positions_matrix <- alignment_results$total_aligned_positions_matrix
number_of_matching_positions_matrix <- alignment_results$number_of_matching_positions_matrix

percent_similarity_matrix <- alignment_results$percent_similarity_matrix

alignment_results_list <- alignment_results$alignment_results_list
```

```
alignment_info_matrix <- alignment_results$alignment_info_matrix

output_directory <- tempdir()

# Save the list of alignment results to an RDS file
saveRDS(alignment_results_list, file.path(output_directory, "alignment_results_list.rds"))

# Save matrices to files
write.table(score_matrix, file.path(output_directory, "score_matrix.txt"), sep = "\t")
average_percent_similarity <- compute_average_similarity_matrix(percent_similarity_matrix)
print(average_percent_similarity)

output_directory <- tempdir()
width_inch <- 8
height_inch <- 6
dpi <- 300

clustering_result <- clustering_percent_similarity(percent_similarity_matrix)

# Extract the dendrogram and clustered data
dend_colored <- clustering_result$dendrogram
clustered_data <- clustering_result$clustered_data
Cluster_SampleID_Percentage <- clustering_result$Cluster_SampleID_Percentage
Cluster_TotalPercentage <- clustering_result$Cluster_TotalPercentage

tiff_file <- file.path(output_directory, "6. hierarchical_clustering_dendrogram_colored.tiff")

# Save the dendrogram as a TIFF image
tiff(tiff_file, width = width_inch, height = height_inch, units = "in", res = dpi)
plot(dend_colored, main = "Colored Hierarchical Clustering Dendrogram")
dev.off()

# Save the clustered data frame to a CSV file
write.csv(clustered_data, file.path(output_directory, "7. clustered_data.csv"), row.names = FALSE)

# Example usage with percent_similarity_matrix
result <- tree_percent_similarity(percent_similarity_matrix)
tree <- result$tree
color_palette <- result$color_palette

tree_newick <- file.path(output_directory, "phylogenetic_tree_nj.nwk")

# Save the phylogenetic tree as a Newick file
# ape::write.tree(tree, file = tree_newick)

# Save the phylogenetic tree as a TIFF image
width_inch <- 8
height_inch <- 6
dpi <- 300

# Set the file name for the TIFF image
tiff_file <- file.path(output_directory, "phylogenetic_tree.tiff")
```

```
# Open the TIFF device
tiff(tiff_file, width = width_inch, height = height_inch, units = "in", res = dpi)

# Plot the phylogenetic tree vertically
plot(tree, main = "Neighbor-Joining Tree", cex = 1, direction = "downward")

# Close the TIFF device
dev.off()

# Set the file name for the TIFF image with rainbow-colored branches
tiff_file_rainbow <- file.path(output_directory, "phylogenetic_tree_rainbow.tiff")

# Open the TIFF device
tiff(tiff_file_rainbow, width = width_inch, height = height_inch, units = "in", res = dpi)

# Plot the phylogenetic tree vertically with rainbow-colored branches
plot(tree, main = "NJ Tree", cex = 1, direction = "downward", tip.color = color_palette)

# Close the TIFF device
dev.off()
```

write_fasta *Writes a data frame to a FASTA file.*

Description

This function takes a data frame with 'SequenceID' and 'Sequence' columns and writes the contents to a FASTA file. Each row in the data frame corresponds to a sequence in the FASTA file, with the 'SequenceID' used as the header line and the 'Sequence' as the sequence line.

Usage

```
write_fasta(data)
```

Arguments

data	A data frame with 'SequenceID' and 'Sequence' columns.

Value

A character vector representing the contents of the FASTA file.

Examples

```
accession_ranges <- list(
  SRU1 = "AJ240966 to AJ240970",
  STU2 = "AB015240 to AB015245",
  WPU13 = "L11934 to L11939",
  INU20 = c("AF277467 to AF277470", "AF333080 to AF333085")
)

# Use the function to expand accession ranges
sam_acc <- expand_accession_ranges(accession_ranges)
print(sam_acc)

# 2 get_sequence_information
accessions_to_query <- sam_acc$accession
seq_info <- get_sequence_information(accessions_to_query, remove_dot_1 = TRUE)
print(seq_info)
result <- preprocess_for_alignment(sam_acc, seq_info)

# Access the resulting data frames
merged_data <- result$merged_data
main_data <- result$main_data
final_data <- result$final_data

data <- final_data

# Call the function
fasta_content <- write_fasta(data)

# Print or use the `fasta_content` as needed
print(fasta_content)

output_directory <- tempdir()
# Specify the output file path
output_file_path <- file.path(output_directory, "output.fasta")

# Open a connection to the output file
output_file <- file(output_file_path, "w")

# Write the content to the file
writeLines(fasta_content, output_file)

# Close the output file
close(output_file)

# Print a message indicating successful file creation
warning("FASTA file has been created and saved at:", output_file_path, "\n")
```

Index