Package ‘doblin’

May 21, 2025

Title 'doblin': Inferring Dominant Clonal Lineages from DNA Barcoding Time-Series

Version 0.1.1

Author Adrian Serohijos [aut, cre],
David Gagné-Leroux [ctb],
Melis Gencel [ctb],
Louis Gauthier [ctb]

Maintainer Adrian Serohijos <`adrian.serohijos@umontreal.ca`>

Description Provides functions to quantify dominant clonal lineages from DNA barcoding time-series data. The package implements clustering of barcode lineage trajectories, based on the assumption that similar temporal dynamics indicate comparable relative fitness. It also identifies persistent clonal lineages across time points. Input data can include lineage frequency tables derived from chromosomal barcoding, mutational libraries, or CRISPR/Cas screens. For more details, see Gagné-Leroux et al. (2024) <doi:10.1101/2024.09.08.611892>.

Imports entropy, gplots, lazyeval, proxy, grid, ggthemes, ggplot2, magrittr, dplyr, ggnewscale, readr, data.table, reshape2, grDevices, stats, imputeTS, dtwclust, purrr, tidyr, TSdist, graphics

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.3.9000

Suggests knitr, rmarkdown, testthat (>= 3.0.0), devtools, ggpubr, optparse, pryr

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Repository CRAN

Date/Publication 2025-05-21 08:50:02 UTC

Contents

`adjust_span` *Recursively adjust the LOESS span to ensure a stable fit*

Description

This function attempts to fit a LOESS (Locally Estimated Scatterplot Smoothing) curve to the provided data using a span parameter that controls the degree of smoothing. If the initial fit produces NA values in key components of the model object, the function recursively increases the span until a stable fit is achieved or the span exceeds 1.

Usage

```
adjust_span(x, y, span)
```

Arguments

`x`	A numeric vector representing the predictor variable (e.g., time or position).
`y`	A numeric vector representing the response variable to be smoothed. Must be non-negative due to log-transformation.
`span`	A numeric value specifying the smoothing parameter for the LOESS fit. Defaults to a low value (e.g., 0.2) and is increased recursively if the fit fails.

Details

This approach avoids requiring the user to manually tune the span, instead adapting to the data distribution and scale. The fit is performed on the log-transformed y values to reduce skewness and avoid numerical instability.

Value

A fitted loess object if a valid fit is achieved.

Examples

```
set.seed(123)
x <- seq(0, 10, length.out = 100)
y <- abs(sin(x)) + rnorm(100, sd = 0.1)

# Fit using recursive span adjustment
fit <- adjust_span(x, y, span = 0.2)
```

`apply_LOESS` *Apply LOESS smoothing to barcode trajectories*

Description

Apply LOESS smoothing to barcode trajectories

Usage

```
apply_LOESS(c_series, output_directory, input_name)
```

Arguments

`c_series`	A dataframe containing `time`, `frequency`, and `cluster` columns.
`output_directory`	A string specifying the directory where plots will be saved.
`input_name`	A string used as the base name for output files (e.g., "replicate1").

Value

A dataframe containing the LOESS-smoothed trajectories for each cluster, with columns `cluster`, `value`, and `time`.

`calculateDiversity` *Calculate diversity indices for a sample*

Description

This is the main function to compute barcode diversity indices for a given sample. It calculates three common diversity measures: species richness (q = 0), Shannon diversity (q = 1), and dominance-based diversity (q = infinity).

Reshapes a long-format data frame into wide format, with lineage IDs as rows and time points as columns. Replaces missing values with zeros.

Calculates the number of lineages with nonzero frequency at each time point.

Calculates the Shannon entropy at each time point and returns its exponential form. This measure considers both the number and evenness of lineages.

Calculates the reciprocal of the most abundant lineage's frequency at each time point. This measure reflects the dominance of the most frequent lineage.

Usage

```
calculate_diversity(input_data)

format_sample(sample)

calculate_q_0(mat)

calculate_q_1(mat)

calculate_q_inf(mat)
```

Arguments

`input_data`	A data frame with columns ID, Time, and Reads. Represents barcode counts per lineage at each time point.
`sample`	A data frame with columns ID, Time, and Reads.
`mat`	A matrix of relative abundances, with IDs as rows and time points as columns.

Details

Internally, the function calls:

- format_sample() to reshape the data
- calculate_q_0(), calculate_q_1(), and calculate_q_inf() to compute each diversity index

Value

A data frame containing three diversity indices over time: q_0 (richness), q_1 (Shannon), and q_inf (dominance).

A wide-format data frame suitable for diversity calculations.

A data frame with one column: q_0.

A data frame with one column: q_1.

A data frame with one column: q_inf.

Examples

```
# Load demo barcode count data (installed with the package)
demo_file <- system.file("extdata", "demo_input.csv", package = "doblin")
input_dataframe <- readr::read_csv(demo_file, show_col_types = FALSE)

# Calculate diversity indices over time
diversity_df <- calculate_diversity(input_dataframe)
```

`fetchTop` *Fetch Top Barcodes by Maximum Frequency*

Description

This function retrieves the top `N_LINEAGES` barcodes from a reshaped dataframe, as returned by `reshapeData()`. The top barcodes are assumed to be located in the first `N_LINEAGES` rows of the reshaped dataframe. This implies that `reshaped_df` should already be sorted by descending maximum frequency for correct behavior.

Usage

```
fetchTop(reshaped_df, N_LINEAGES)
```

Arguments

`reshaped_df`	A dataframe returned by `reshapeData()`. Only the first four columns are used.
`N_LINEAGES`	An integer specifying the number of barcodes to retrieve.

Value

A dataframe containing the top `N_LINEAGES`. Used for selecting dominant lineages for plotting or analysis.

Examples

```
# Load demo barcode count data (installed with the package)
demo_file <- system.file("extdata", "demo_input.csv", package = "doblin")
input_dataframe <- readr::read_csv(demo_file, show_col_types = FALSE)

# Reshape & sort the data
reshaped_df <- reshapeData(input_dataframe)

# Fetch the top 10 most abundant barcodes
top_barcodes <- fetchTop(reshaped_df, N_LINEAGES = 10)
```

---

filterData *Filter Lineage Data for Clustering*

Description

This function filters lineage frequency data to retain only dominant and persistent barcodes suitable for clustering. It removes barcodes that do not meet a specified minimum mean frequency and a minimum number of time points with non-zero frequency. The function saves two CSV files: one with all original barcodes and one with the filtered set.

Usage

```
filterData(
  input_df,
  freq_threshold,
  time_threshold,
  output_directory,
  input_name
)
```

Arguments

`input_df`	A data frame containing the input data. It must have columns `ID`, `Time`, and `Reads`.
`freq_threshold`	A numeric value specifying the minimum mean frequency required to retain a barcode.
`time_threshold`	An integer specifying the minimum number of time points where the barcode's frequency is non-zero.
`output_directory`	A string specifying the directory where plots will be saved.
`input_name`	A string used as the base name for output files (e.g., "replicate1").

Value

A data frame containing the ID, relative frequency at each time point, mean frequency, and number of non-zero time points for each retained barcode.

Examples

```
# Load demo barcode count data (installed with the package)
demo_file <- system.file("extdata", "demo_input.csv", package = "doblin")
input_dataframe <- readr::read_csv(demo_file, show_col_types = FALSE)

# Apply filtering to retain dominant and persistent barcodes
filtered_df <- filterData(
  input_df = input_dataframe,
  freq_threshold = 0.00005,
  time_threshold = 5,
  output_directory = tempdir(),
  input_name = "demo"
)
```

`filterHC` *Filter Hierarchical Clusters Based on Size and Dominance*

Description

This function filters the results of hierarchical clustering by retaining only clusters that contain at least `n_members` unique lineages. To avoid excluding potentially dominant but small clusters, the user may also provide a minimum average frequency threshold to retain small clusters that include a dominant member.

Usage

```
filterHC(
  series_filtered,
  clusters,
  n_members,
  min_freq_ignored_clusters = NULL
)
```

Arguments

`series_filtered`	A data frame preprocessed using `filterData()`, containing lineage frequencies and metadata.
`clusters`	A data frame containing hierarchical clustering assignments (e.g., from `cutree()`), possibly across multiple thresholds.
`n_members`	An integer specifying the minimum number of members (lineages) required for a cluster to be retained.
`min_freq_ignored_clusters`	Optional. A numeric value specifying the minimum average frequency required to retain small clusters (i.e., those with fewer than `n_members`). If `NULL`, small clusters are not rescued.

Value

A data frame containing the filtered clusters, including both large clusters and optionally small clusters with at least one dominant member (based on the `min_freq_ignored_clusters` threshold).

Examples

```
# Load demo barcode count data (installed with the package)
demo_file <- system.file("extdata", "demo_input.csv", package = "doblin")
input_dataframe <- readr::read_csv(demo_file, show_col_types = FALSE)

# Filter data to retain dominant and persistent barcodes
filtered_df <- filterData(
  input_df = input_dataframe,
  freq_threshold = 0.00005,
  time_threshold = 5,
  output_directory = tempdir(),
  input_name = "demo"
)

# Perform hierarchical clustering using Pearson correlation
cluster_assignments <- performHClustering(
  filtered_data = filtered_df,
  agglomeration_method = "average",
  similarity_metric = "pearson",
  output_directory = tempdir(),
  input_name = "demo",
  missing_values = "pairwise.complete.obs",
  dtw_norm = NULL
)

# Filter clusters: keep only clusters with at least 8 members.
filtered_clusters <- filterHC(
  series_filtered = filtered_df,
  clusters = cluster_assignments,
  n_members = 8,
  min_freq_ignored_clusters = 0.0001
)
```

`performHClustering` *Perform Hierarchical Clustering on Barcoded Lineages*

Description

This function performs hierarchical clustering on time-series data representing barcoded lineages. A distance matrix is computed using either Pearson correlation or Dynamic Time Warping (DTW), and hierarchical clustering is applied using a specified agglomeration method. A dendrogram and heatmap are generated for visual inspection. If no threshold is specified, clusters are computed for all possible thresholds between 0.1 and the maximum tree height.

Usage

```
performHClustering(
  filtered_data,
  agglomeration_method,
  similarity_metric,
  output_directory,
  input_name,
  missing_values = NULL,
  dtw_norm = NULL
)
```

Arguments

`filtered_data`	A data frame preprocessed with `filterData()`, containing filtered lineage frequencies.
`agglomeration_method`	A character string specifying the agglomeration method (e.g., `"ward.D"`, `"complete"`).
`similarity_metric`	A character string specifying the similarity metric (`"pearson"` or `"dtw"`).
`output_directory`	A string specifying the directory where plots will be saved.
`input_name`	A string used as the base name for output files (e.g., "replicate1")
`missing_values`	Optional. A character string specifying how missing values should be handled in Pearson correlation (e.g., `"pairwise.complete.obs"`).
`dtw_norm`	Optional. A character string specifying the norm to use with DTW distance ("L1" for Manhattan, "L2" for Euclidean). Required if `similarity_metric = "dtw"`.

Value

A data frame with clustering assignments at multiple thresholds (columns named by height).

Examples

```
# Load demo barcode count data (installed with the package)
demo_file <- system.file("extdata", "demo_input.csv", package = "doblin")
input_dataframe <- readr::read_csv(demo_file, show_col_types = FALSE)

# Filter data to retain dominant and persistent barcodes
filtered_df <- filterData(
  input_df = input_dataframe,
  freq_threshold = 0.00005,
  time_threshold = 5,
  output_directory = tempdir(),
  input_name = "demo"
)

# Perform hierarchical clustering using Pearson correlation
```

```
cluster_assignments <- performHClustering(
  filtered_data = filtered_df,
  agglomeration_method = "average",
  similarity_metric = "pearson",
  output_directory = tempdir(),
  input_name = "demo",
  missing_values = "pairwise.complete.obs",
  dtw_norm = NULL
)
```

plotClusterLog10 — *Plot individual barcode frequencies (log10) for a single cluster*

Description

Plots all barcodes in a cluster on a log10 y-scale, along with the LOESS-smoothed average trajectory.

Usage

```
plotClusterLog10(
  df,
  cluster,
  color,
  tf,
  effective.breaks,
  output_directory,
  input_name
)
```

Arguments

`df`	A dataframe containing barcode frequencies in a single cluster.
`cluster`	The cluster ID (numeric or character).
`color`	A color code to use for the cluster.
`tf`	A dataframe containing the LOESS-smoothed trajectory for the cluster.
`effective.breaks`	A vector of time points used as breaks on the x-axis.
`output_directory`	A string specifying the directory where plots will be saved.
`input_name`	A string used as the base name for output files (e.g., "replicate1").

Value

A ggplot object showing the log10-transformed barcode frequencies and the LOESS-smoothed average trajectory for a single cluster.

plotClustersAndLoess *Plot the log10-transformed barcode frequencies and the moving averages (LOESS)*

Description

This file contains multiple functions. The main function is: plot_clusters_and_loess() and it uses plotClusterLog10() and apply_LOESS(). In plot_clusters_and_loess(), we plot the log10-transformed barcode frequencies contained in all selected clusters, we compute a moving average per cluster and group them in a plot. We also write the files associated with these two plots.

Usage

```
plotClustersAndLoess(selected_clusters, output_directory, input_name)
```

Arguments

`selected_clusters`	A dataframe containing the clusters from a hierarchical clustering for a specific threshold
`output_directory`	A string specifying the directory where plots will be saved.
`input_name`	A string used as the base name for output files (e.g., "replicate1").

Value

No return value. This function saves plots and CSV files related to barcode cluster dynamics.

Examples

```
# Load demo barcode count data (installed with the package)
demo_file <- system.file("extdata", "demo_input.csv", package = "doblin")
input_dataframe <- readr::read_csv(demo_file, show_col_types = FALSE)

# Filter data to retain dominant and persistent barcodes
filtered_df <- filterData(
  input_df = input_dataframe,
  freq_threshold = 0.00005,
  time_threshold = 5,
  output_directory = tempdir(),
  input_name = "demo"
)

# Perform hierarchical clustering using Pearson correlation
cluster_assignments <- performHClustering(
  filtered_data = filtered_df,
  agglomeration_method = "average",
  similarity_metric = "pearson",
  output_directory = tempdir(),
  input_name = "demo",
  missing_values = "pairwise.complete.obs",
  dtw_norm = NULL
)

# Filter clusters to retain only those with at least 8 members,
# unless they contain a dominant lineage
filtered_clusters <- filterHC(
  series_filtered = filtered_df,
  clusters = cluster_assignments,
  n_members = 8,
  min_freq_ignored_clusters = 0.0001
)


# Plot log10-transformed barcode frequencies and smoothed LOESS average per cluster
plotClustersAndLoess(
  selected_clusters = filtered_clusters,
  output_directory = tempdir(),
  input_name = "demo"
)
```

plotDiversity *Plot diversity dynamics over time*

Description

This function plots the diversity of barcoded populations across generations. A multi-panel EPS figure is saved, showing one panel per diversity order.

Usage

```
plotDiversity(dataframe, output_directory, input_name)
```

Arguments

`dataframe`	A data frame containing barcode diversities with columns for generations and diversity metrics.
`output_directory`	A string specifying the directory where plots will be saved.
`input_name`	A string used as the base name for output files (e.g., "replicate1").

Value

A faceted ggplot object (invisible). The function also saves the figure to an EPS file.

Examples

```
# Load demo barcode count data (installed with the package)
demo_file <- system.file("extdata", "demo_input.csv", package = "doblin")
input_dataframe <- readr::read_csv(demo_file, show_col_types = FALSE)

# Calculate diversity indices over time
diversity_df <- calculate_diversity(input_dataframe)

# Plot and save diversity figure
plotDiversity(
  dataframe = diversity_df,
  output_directory = tempdir(),
  input_name = "demo"
)
```

`plotDynamics` *Plot barcode dynamics*

Description

This function plots the dynamics of barcode frequencies over time, using either linear-scale area plots, logarithmic-scale line plots, or both. Only the most frequent barcodes are colored.

Usage

```
plotDynamics(
  reshaped_df,
  colored_topFreq_df,
  min_freq_threshold,
  plot_model,
  output_directory,
  input_name
)
```

Arguments

`reshaped_df`	A dataframe produced by `reshapeData()`, containing barcode frequencies over time.
`colored_topFreq_df`	A dataframe with top barcodes and their assigned color hex codes and max frequencies.
`min_freq_threshold`	A numeric threshold; barcodes with max frequency below this are colored gray.
`plot_model`	One of `"linear"`, `"logarithmic"`, or `"both"` to specify the plot type(s).
`output_directory`	A string specifying the directory where plots will be saved.
`input_name`	A string used as the base name for output files (e.g., "replicate1").

Value

No return value. Depending on the `plot_model` parameter:

- Saves a linear-scale area plot (`_area.jpg`) showing the dynamics of barcode frequencies over time.
- Saves a logarithmic-scale line plot (`_line.eps`) highlighting prominent barcodes across time.

Examples

```
# Load demo barcode count data
demo_file <- system.file("extdata", "demo_input.csv", package = "doblin")
input_dataframe <- readr::read_csv(demo_file, show_col_types = FALSE)

# Reshape and extract top lineages
reshaped_df <- reshapeData(input_dataframe)
top_barcodes <- fetchTop(reshaped_df, N_LINEAGES = 10)

# Load color list and assign hex codes
color_file <- system.file("extdata", "top_colors2.csv", package = "doblin")
color_df <- readr::read_csv(color_file, show_col_types = FALSE)
color_df <- color_df[1:nrow(top_barcodes), ]
colored_top <- cbind(top_barcodes, color_df)

# Plot dynamics
plotDynamics(
  reshaped_df = reshaped_df,
  colored_topFreq_df = colored_top,
  min_freq_threshold = 0.001,
  plot_model = "both",
  output_directory = tempdir(),
  input_name = "demo"
)
```

`plotHCQuantification` *Quantify and Visualize Hierarchical Clustering Results*

Description

This script contains several functions to help quantify and visualize the results of hierarchical clustering on barcode time-series data. The main function is `plotHCQuantification()`, which computes a LOESS-smoothed average of barcode frequencies per cluster and evaluates inter-cluster distances across different clustering thresholds.

The melt_dist() function takes a distance matrix and converts it into a long-format data frame where each row corresponds to a unique pair of elements and their associated distance. It essentially "melts" the lower triangle of the matrix into a tidy format, which is useful for plotting or further analysis.

Applies LOESS smoothing to barcode frequencies within each cluster over time, using only the persistent barcodes (those present at the last time point). Clusters are re-ranked within each threshold based on their average final frequency.

Usage

```
plotHCQuantification(clusters_filtered, output_directory, input_name)

melt_dist(dist, order = NULL, dist_name = "dist")

applyLOESS(clusters_filtered)
```

Arguments

`clusters_filtered`	A data frame filtered by `filterHC()`, containing `cluster`, `cutoff`, `Time`, and `Frequency` columns.
`output_directory`	A string specifying the directory where plots will be saved.
`input_name`	A string used as the base name for output files (e.g., "replicate1").
`dist`	A distance matrix (typically a result of a distance computation).
`order`	Optional character vector indicating the order of row/column names to rearrange the matrix before melting.
`dist_name`	A string naming the distance variable in the resulting data frame. Default is "dist".

Value

No return value. This function saves a plot and a CSV file containing the smallest inter-cluster distances per threshold.

A data frame with columns: `iso1`, `iso2`, and the specified distance column.

A data frame with smoothed values for each cluster and time point: columns include `cluster`, `cutoff`, `model`, and `time`.

Examples

```
# Load demo barcode count data (installed with the package)
demo_file <- system.file("extdata", "demo_input.csv", package = "doblin")
input_dataframe <- readr::read_csv(demo_file, show_col_types = FALSE)

# Filter data to retain dominant and persistent barcodes
filtered_df <- filterData(
  input_df = input_dataframe,
  freq_threshold = 0.00005,
  time_threshold = 5,
  output_directory = tempdir(),
  input_name = "demo"
)
```

```
# Perform hierarchical clustering using Pearson correlation
cluster_assignments <- performHClustering(
  filtered_data = filtered_df,
  agglomeration_method = "average",
  similarity_metric = "pearson",
  output_directory = tempdir(),
  input_name = "demo",
  missing_values = "pairwise.complete.obs",
  dtw_norm = NULL
)

# Filter clusters to retain only those with at least 8 members,
# unless they contain a dominant lineage
filtered_clusters <- filterHC(
  series_filtered = filtered_df,
  clusters = cluster_assignments,
  n_members = 8,
  min_freq_ignored_clusters = 0.0001
)

# Quantify and visualize clustering quality across thresholds
plotHCQuantification(
  clusters_filtered = filtered_clusters,
  output_directory = tempdir(),
  input_name = "demo"
)
```

reshapeData — *Reshape Barcode Abundance Data to Frequency Format*

Description

Transforms raw barcode abundance data into a tidy long-format data frame, computing summary statistics for each barcode (ID), including maximum, initial, final, and average frequencies across time points.

Usage

```
reshapeData(input_data)
```

Arguments

`input_data`	A data frame with exactly three columns: `ID` (character or factor), `Time` (numeric), and `Reads` (numeric). Each row corresponds to a measurement for one barcode at one time point.

Details

This function expects a data frame with three columns: `ID`, `Time`, and `Reads`. Frequencies are computed by normalizing the `Reads` across all barcodes for each time point.

Value

A tidy data frame with columns: `ID`, `max`, `start`, `final`, `mean`, `Time`, and `Frequency`. Frequencies are normalized across all barcodes per time point. The result is ordered by decreasing `max` frequency.

Examples

```
# Load demo barcode count data (installed with the package)
demo_file <- system.file("extdata", "demo_input.csv", package = "doblin")
input_dataframe <- readr::read_csv(demo_file, show_col_types = FALSE)

# Reshape data to long-format with normalized frequencies +
# sort data by descending maximum frequency
reshaped_df <- reshapeData(input_dataframe)
```

`theme_Matrix` *Theme for matrix-style plots (e.g. heatmaps or abundance matrices)*

Description

Theme for matrix-style plots (e.g. heatmaps or abundance matrices)

Usage

```
theme_Matrix(base_size = 24, base_family = "Arial")
```

Arguments

`base_size`	Base font size.
`base_family`	Base font family.

Value

A ggplot2 theme object.

theme_Publication *Custom ggplot2 theme for publication-style figures*

Description

Custom ggplot2 theme for publication-style figures

Usage

```
theme_Publication(base_size = 24, base_family = "Arial", aspect.ratio = 0.75)
```

Arguments

`base_size`	Base font size.
`base_family`	Base font family.
`aspect.ratio`	Aspect ratio of the plot.

Value

A ggplot2 theme object.

theme_Publication_noYaxis *Custom ggplot2 theme with no y-axis title*

Description

Custom ggplot2 theme with no y-axis title

Usage

```
theme_Publication_noYaxis(
  base_size = 24,
  base_family = "Arial",
  aspect.ratio = 0.75
)
```

Arguments

`base_size`	Base font size.
`base_family`	Base font family.
`aspect.ratio`	Aspect ratio of the plot.

Value

A ggplot2 theme object.

Index