Package: profrep (via r-universe)

August 31, 2024

Title Profile Repeatability

Version 1.0.0

Description Calculates profile repeatability for replicate stress response curves, or similar time-series data. Profile repeatability is an individual repeatability metric that uses the variances at each timepoint, the maximum variance, the number of crossings (lines that cross over each other), and the number of replicates to compute the repeatability score. For more information see Reed et al. (2019) <doi:10.1016/j.ygcen.2018.09.015>.

Depends R (>= 2.10)

License MIT + file LICENSE

URL https://ubeattie.github.io/profrep/

BugReports https://github.com/ubeattie/profrep/issues

Encoding UTF-8

Roxygen list(markdown=TRUE)

RoxygenNote 7.2.3

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports stats

LazyData true

VignetteBuilder knitr

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

RemoteUrl https://github.com/ubeattie/profrep

RemoteRef HEAD

RemoteSha 6de9e06d64e3690e5b291f55be1efca2388b8804

Contents

`calculate_crossovers` *Calculate the Number of Crossovers*

Description

Calculate the Number of Crossovers

Usage

```
calculate_crossovers(individual_df, n_trials, n_replicates)
```

Arguments

`individual_df`	A data frame containing the individual dataset.
`n_trials`	The total number of trials in the dataset (the number of rows)
`n_replicates`	The total number of replicates in each trial (the number of columns - 2)

Details

This function calculates the number of crossovers in a dataset by comparing the values of replicates across multiple trials. It assumes that missing values (NAs) have been interpolated using the `clean_data` function.

Value

The number of crossovers detected in the dataset.

See Also

`clean_data` for information on data cleaning.

Examples

```
data <- matrix(
   c(
     1, 60, 1, 2, 3, 4, 5,   # No NA values
     1, 90, 9, NA, 4, NA, 2,  # NA Values in row
     1, 120, 3, 6, NA, NA, 9  # Consecutive NA values
    ),
    nrow = 3,
    byrow=TRUE
)
n_trials <- nrow(data)
n_replicates <- ncol(data) - 2
crossovers <- calculate_crossovers(data, n_trials, n_replicates)
cat("Number of crossovers:", crossovers, "\n")
```

---

clean_data *Clean Data by Interpolating Missing Values*

Description

Clean Data by Interpolating Missing Values

Usage

```
clean_data(data, n_trials, n_replicates)
```

Arguments

`data`	A data frame containing the dataset to be cleaned.
`n_trials`	The total number of rows in the dataset.
`n_replicates`	The total number of replicate columns in each row.

Details

This function cleans a dataset by interpolating missing values in the replicate columns of each row using neighboring values. If the data frame ends in null values (the last columns are nulls), it will extrapolate from the last value. If the first value is null, it will loop around and pull from the last replicate to perform the interpolation between the last replicate and the second replicate.

Value

A cleaned data frame with missing values interpolated.

See Also

`find_next_good_datapoint` for details on the interpolation process.

Examples

```
my_data <- matrix(
   c(
     1, 60, 1, 2, 3, 4, 5,   # No NA values
     1, 90, 9, NA, 4, NA, 2,  # NA Values in row
     1, 120, 3, 6, NA, NA, 9  # Consecutive NA values
    ),
    nrow = 3,
    byrow=TRUE
 )
cleaned_data <- clean_data(my_data, n_trials = 3, n_replicates = 5)
print(my_data)
print(cleaned_data)
```

`do_ordering` *Score and Order Data*

Description

Score and Order Data

Usage

```
do_ordering(
  n_trials,
  id_list,
  df_list,
  n_replicates,
  verbose = FALSE,
  sort = TRUE
)
```

Arguments

`n_trials`	The number of rows an individual sample will have.
`id_list`	The list of unique individual or sample names
`df_list`	The list of data frames per unique individual
`n_replicates`	The number of replicates in the study.
`verbose`	A boolean parameter the defaults to FALSE. Determines whether messages are printed.
`sort`	A boolean parameter that defaults to TRUE. If TRUE, sorts the returned data frame by score. If FALSE, returns the data in the individual order it was provided in

Details

Performs the ordering of input data by scoring each individual data frame.

The main function of the package, this will send each individuals data out for scoring. Then, when all scores are computed, it will order the result data frame by score and assign a rank.

Ranks are assigned with ties allowed - if N individuals have a tie, their rank is averaged. For example, if the max score is 1, and two individuals have that score, their rank is 1.5

Value

Returns a data frame of the results, in the following form:

```
    - Column 1: "individual" - the unique identifier of an individual or sample
    - Column 2: "n_crossings" - the calculated number of crossings.
  - Column 3: "max_variance" - the maximum of the variances of the replicate measurements at a single ti
  - Column 4: "ave_variance" - the average of the variances of the replicate measurements at a single ti
  - Column 5: "base_score" - the original, unnormalized profile repeatability score. Smaller numbers r
  - Column 6: "final_score" - the base score, normalized by the sigmoid function. Constrained to be bet
  - Column 7: "rank" - the calculated ranking of the individual or sample, against all other individual
```

Examples

```
df <- data.frame(
    col_a = c('A', 'A', 'B', 'B'),
    col_b = c(5, 15, 5, 15),
    col_c = c(5, 10, 1, 2),
    col_d = c(10, 15, 3, 4)
  )
id_list <- unique(df[, 1])
individuals <- list()
for (i in 1:length(id_list)) {
  individuals[[i]] <- df[df[, 1] == id_list[i], ]
}
ret_df <- do_ordering(n_trials=2, id_list=id_list, df_list=individuals, n_replicates=2)
print(ret_df)
```

---

example_two_point_data

Example Data: Two Point Data

Description

An example of data that one would perform profile repeatability on. Consists of 9 individual animals, with corticosterone data taken at 2 timepoints (n_trials = 2), baseline (time = 3) and stress-induced (time = 30). Then, there are 28 replicate columns.

Usage

```
example_two_point_data
```

Format

`example_two_point_data`**:**

A dataframe with 10 rows and 30 columns:

Animal The animal name/unique identifier

Time The time of the measurement, in days.

SD.DR The name of the replicate column.

Source

This data was extracted from Romero & Rich 2007 (Comp Biochem. Physiol. Part A Mol. Integr. Physiol. 147, 562-568. https://doi.org/10.1016/j.cbpa.2007.02.004)

`find_next_good_datapoint` *What Is the Next Non-Null Data Point?*

Description

What Is the Next Non-Null Data Point?

Usage

```
find_next_good_datapoint(data_row, index, n_replicates)
```

Arguments

`data_row`	A numeric vector representing the data row.
`index`	The index of the current data point.
`n_replicates`	The total number of replicates (length of the row)

Details

Given a data row, an index, and the number of replicates (the number of elements in the row), this function finds the next good data point in the row.

A good data point is a non-missing value (not NA) with a non-empty string.

Value

The next good data point or -999 if none is found.

Examples

```
data_row <- c(NA, 3, 2, NA, 5)
index <- 1
n_replicates <- 5
find_next_good_datapoint(data_row, index, n_replicates) # expect 3
```

`get_vars` *Calculate Group Variance*

Description

Calculate Group Variance

Usage

```
get_vars(individual_array, n_replicates)
```

Arguments

`individual_array`	The array of data for an individual
`n_replicates`	The number of replicate groups

Details

For the given individual array, for all rows of times, computes the variance in values over replicates.

Returns these variances, sum of all values (for all times and replicates), sum of all these values squared, and the number of values.

Value

A list, where the elements are:

1. `variances`: A vector of the variances of the sample
2. `total_sum`: The sum of all the measurements in the sample
3. `ssq`: The sum of all the squares of the measurements in the sample
4. `num_measurements`: The total number of measurements in the sample that are not null

Examples

```
arr <- data.frame(
  individual=c("a", "a"),
  time=c(5, 15),
  col_a=c(1, 2),
  col_b=c(2, 3)
)
variance_return <- get_vars(individual_array=arr, n_replicates=2)
print(variance_return)
```

profrep *Perform Profile Repeatability*

Description

Perform Profile Repeatability

Usage

```
profrep(df, n_timepoints, sort = TRUE, verbose = FALSE)
```

Arguments

df	The input data frame, of minimum shape 3 rows by 4 columns. This can be read in from a csv or another data frame stored in memory. It is assumed that the data frame is of the following structure: Column 1 is the unique identifier of an individual animal or sample Column 2 is the time of the sample Column 3-N are the columns of replicate data. Row 1 is assumed to be header strings for each column.
n_timepoints	The number of rows an individual sample will have. For example, if the replicates were collected for individual 1 at times 15 and 30, for replicates A and B, the data frame would look like:

```
| id | time | A | B |
|:--:|:----:|:-:|:-:|
| 1  | 15 | 1 | 2 |
| 1  | 30 | 3 | 4 |
```

sort	A boolean parameter that defaults to TRUE. If TRUE, sorts the returned data frame by score. If FALSE, returns the data in the individual order in which it was provided.
verbose	A boolean parameter that defaults to FALSE. Determines whether messages are printed.

Details

Calculates the profile repeatability measure of the input data according to the method in Reed et al., 2019, J. Gen. Comp. Endocrinol. (270).

Value

Returns a data frame of the results, in the following form:

- Column 1: "individual" - the unique identifier of an individual or sample
- Column 2: "n_crossings" - the calculated number of crossings.
- Column 3: "max_variance" - the maximum of the variances of the replicate measurements at a single time for the individual or sample.

- Column 4: "ave_variance" - the average of the variances of the replicate measurements at a single time for the individual or sample.
- Column 5: "base_score" - the original, unnormalized profile repeatability score. Smaller numbers rank higher.
- Column 6: "final_score" - the base score, normalized by the sigmoid function. Constrained to be between 0 and 1. Scores closer to 1 rank higher.
- Column 7: "rank" - the calculated ranking of the individual or sample, against all other individuals or samples in the data set.

See Also

`do_ordering` for the main data processing function.

`calculate_crossovers` for how the number of crossings are calculated.

`score_individual_df` for how the score is calculated for an individual or sample.

`clean_data` for how missing replicate values are handled.

Examples

```
test_data <- profrep::example_two_point_data
results <- profrep::profrep(df=test_data, n_timepoints=2)
print(results)
```

---

retrieve_good_data *Retrieve Indices of Non-Missing Data for a Specific Time Point*

Description

This function retrieves the indices of non-missing data values at a specific time point from an individual array.

Usage

```
retrieve_good_data(individual_array, t, n_replicates)
```

Arguments

`individual_array`	A data matrix or data frame representing individual data, where rows correspond to time points and columns correspond to replicates and variables.
`t`	The time point for which you want to retrieve non-missing data indices.
`n_replicates`	The number of replicates in the data matrix.

Value

A numeric vector containing the indices of non-missing data values at the specified time point `t`. If there are no non-missing values or only one non-missing value, `NULL` is returned.

See Also

`which` function for finding the indices of non-missing values.

Examples

```
# Example usage:
individual_data <- matrix(c(NA, 2, NA, 4, 5, NA), nrow = 1)
retrieve_good_data(individual_data, t = 1, n_replicates = 3)
```

---

score_dfs *Compute Profile Repeatability Score*

Description

Compute Profile Repeatability Score

Usage

```
score_dfs(id_list, df_list, n_replicates, n_trials, verbose = FALSE)
```

Arguments

id_list	The list of the names of the individuals
df_list	A list of data frames, each of which correspond to one of the names in the individual list
n_replicates	The number of replicate columns (number of columns in a df in df_list)
n_trials	The number of trials per individual (number of rows in a df in df_list)
verbose	A boolean parameter the defaults to FALSE. Determines whether messages are printed.

Details

Works on multiple elements of data.

Splits the data into the data frame for a particular individual from the id_list, then calculates metrics to compute the profile repeatability score. Returns a data frame with the individuals name and the score.

Value

A dataframe of the calculated metrics. The column structure is as follows:

```
    - Column 1: "individual" - the unique identifier of an individual or sample
    - Column 2: "n_crossings" - the calculated number of crossings.
  - Column 3: "max_variance" - the maximum of the variances of the replicate measurements at a single ti
  - Column 4: "ave_variance" - the average of the variances of the replicate measurements at a single ti
  - Column 5: "base_score" - the original, unnormalized profile repeatability score. Smaller numbers r
  - Column 6: "final_score" - the base score, normalized by the sigmoid function. Constrained to be bet
```

Examples

```
df <- data.frame(
    col_a = c('A', 'A', 'B', 'B'),
    col_b = c(5, 15, 5, 15),
    col_c = c(5, 10, 1, 2),
    col_d = c(10, 15, 3, 4)
  )
id_list <- unique(df[, 1])
individuals <- list()
for (i in 1:length(id_list)) {
  individuals[[i]] <- df[df[, 1] == id_list[i], ]
}
ret_df <- score_dfs(id_list=id_list, df_list=individuals, n_replicates=2, n_trials=2)
print(ret_df)
```

---

score_individual_df *Score an Individual Data Frame*

Description

Score an Individual Data Frame

Usage

```
score_individual_df(
  individual_df,
  n_trials,
  n_replicates,
  max_variance,
  variance_set
)
```

Arguments

individual_df	A data frame containing individual data.
n_trials	The total number of trials in the data frame.
n_replicates	The total number of replicates in each trial.
max_variance	The maximum allowed variance value.
variance_set	A vector of variance values.

Details

This function calculates a score for an individual data frame based on various factors, including the number of crossovers, maximum variance, and a set of variances.

The score is computed as follows:

- It factors in the number of crossovers using a scaling factor.
- It considers the maximum variance value in the variance set.
- It adds a component based on the average of variance values.
- It includes a scaled component of the number of crossovers.

Value

A list calculated for the individual data frame. Contains two values:

1. n_crossings: The number of crossover events in the data.
2. base_score: The un-normalized profile repeatability score for the data.

See Also

`calculate_crossovers` for information on crossovers calculation.

Examples

```
arr <- data.frame(
  individual=c("a", "a"),
  time=c(5, 15),
  col_a=c(1, 2),
  col_b=c(2, 3)
)
variance_set <- c(0.5, 0.5)
max_variance <- 0.5
score_list <- score_individual_df(
  individual_df=arr,
  n_trials=2,
  n_replicates=2,
  max_variance=max_variance,
  variance_set=variance_set
)
print(score_list)
```

---

sigmoid *Calculates the sigmoid function of the input*

Description

Calculates the sigmoid function of the input

Usage

```
sigmoid(float)
```

Arguments

`float`	A float number

Value

A float number which is the result of the sigmoid function

Examples

```
sigmoid(0)
sigmoid(2)
```

`sparrow_repeatability_three_point` *Example Data: Sparrow Repeatability (3 Point Data)*

Description

An example of data that one would perform profile repeatability on. Consists of 12 individual animals, with corticosterone data taken at 3 times (n_trials = 3), baseline (time = 0) and two stress-induced (time = 15 and 30). Then, there are 10 replicate columns. This example also shows what happens when there are null data records for some individuals.

Usage

```
sparrow_repeatability_three_point
```

Format

`sparrow_repeatability_three_point`**:**

A dataframe with 36 rows and 12 columns:

Animal The animal name/unique identifier

TIME The time of the measurement, in days

LD.500 The name of the replicate column

Source

This data was extracted from Rich & Romero 2001 (J. Comp. Physiol. Part B Biochem. Syst. Environ. Physiol. 171, 543-647. https://doi.org/10.1007/s003600100204)

synthetic_data_four_point

Example Data: Synthetic 4-Point Data

Description

An example of data that one would perform profile repeatability on. The data is synthetic data created for testing purposes and is designed to span a range of perceived repeatability scores. Consists of 11 individual animals, with data taken at 4 times (n_trials = 4), baseline (time = 0) and three stress-induced (time = 15, 30, and 45). Then, there are four replicate columns. Replicate column names refer to sample tests performed on the animal.

Usage

```
synthetic_data_four_point
```

Format

synthetic_data_four_point:

A dataframe with 44 rows and 6 columns:

Animal The animal name/unique identifier

TIME The time of the measurement (unit not important)

A The (unimportant) name of a replicate column.

B The (unimportant) name of a replicate column.

C The (unimportant) name of a replicate column.

D The (unimportant) name of a replicate column.

Source

Data created for testing purposes by Reed et al., 2019 (Gen. Comp. Endocrinol. 270, 1-9. https://doi.org/10.1016/j.ygcen.2018.09.015)

Index