Package: utile.tools (via r-universe)

October 10, 2024

Title Summarize Data for Publication

Version 0.3.0

Description Tools for formatting and summarizing data for outcomes research.

License LGPL (>= 2)

URL https://efinite.github.io/utile.tools/

BugReports https://github.com/efinite/utile.tools/issues

Encoding UTF-8

Depends R (>= 3.4.0)

Imports lubridate, purrr, vctrs

Suggests survival, dplyr, ggplot2

RoxygenNote 7.2.3

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

RemoteUrl https://github.com/efinite/utile.tools

RemoteRef HEAD

RemoteSha d059e684da40da3adb8c080fc945ffd661d756ba

Contents

`calc_chunks` *Calculate data chunk indices*

Description

Calculates chunk indices of a data object for a given chunk size (number of items per chunk).

Usage

```
calc_chunks(x, size = 10, reverse = FALSE)
```

Arguments

`x`	A data frame or vector.
`size`	An integer. The number of items (e.g. rows in a tibble) that make up a given chunk. Must be a positive integer. Caps out at data maximum.
`reverse`	A logical. Calculate chunks from back to front.

Value

An iterable list of row indices for each chunk of data.

Examples

```
# Create chunk map for a data frame
chunks <- calc_chunks(mtcars, size = 6)

# Iterate through chunks of data
for (chunk in chunks) print(paste0(rownames(mtcars[chunk,]), collapse = ', '))
```

`calc_duration` *Calculate durations of time*

Description

Calculates the duration of time between two provided date objects. Supports vectorized data (i.e. `dplyr::mutate()`).

Usage

```
calc_duration(x, y, units = NULL)
```

Arguments

x	A date or datetime. The start date(s)/timestamp(s).
y	A date or datetime. The end date(s)/timestamp(s).
units	A character. Units of the returned duration (i.e. 'seconds', 'days', 'years').

Value

If 'units' specified, returns numeric. If 'units' unspecified, returns an object of class 'Duration'.

Note

Supports multiple calculations against a single time point (i.e. multiple start dates with a single end date). Note that start and end must otherwise be of the same length.

When the start and end dates are of different types (i.e. x = date, y = datetime), a lossy cast will be performed which strips the datetime data of its time components. This is done to avoid an assumption of more time passing that would otherwise come with casting the date data to datetime.

Examples

```
library(lubridate)
library(purrr)

# Dates -> duration in years
calc_duration(
  x = mdy(map_chr(sample(1:9, 5), ~ paste0('01/01/199', .x))),
  y = mdy(map_chr(sample(1:9, 5), ~ paste0('01/01/200', .x))),
  units = 'years'
)

# datetimes -> durations
calc_duration(
  x = mdy_hm(map_chr(sample(1:9, 5), ~ paste0('01/01/199', .x, ' 1', .x, ':00'))),
  y = mdy_hm(map_chr(sample(1:9, 5), ~ paste0('01/01/200', .x, ' 0', .x, ':00')))
)

# Mixed date classes -> durations
calc_duration(
  x = mdy(map_chr(sample(1:9, 5), ~ paste0('01/01/199', .x))),
  y = mdy_hm(map_chr(sample(1:9, 5), ~ paste0('01/01/200', .x, ' 0', .x, ':00')))
)
```

---

chunk_data_ *Break data into chunks*

Description

Creates a factory function which returns a different chunk of a given data object with each function call.

Usage

```
chunk_data_(x, size = 10, reverse = FALSE)
```

Arguments

x	A data frame or vector.
size	An integer. The number of items (e.g. rows in a tibble) that make up a given chunk. Must be a positive integer.
reverse	A logical. Calculate chunks from back to front.

Value

A factory function which returns a chunk of data from the provided object with each call. Once all data has been returned, function returns NULL perpetually.

Examples

```
# Create chunk factory function
chunked_data <- chunk_data_(mtcars, size = 6)

# Chunk #1 (rows 1-6)
paste0(rownames(chunked_data()), collapse = ', ')

# Chunk #2 (rows 7-12)
paste0(rownames(chunked_data()), collapse = ', ')
```

---

cusum_failure *Cumulative Sum of Failures*

Description

Calculates the cumulative sum of failures for a series of procedures which can be used to create CUSUM charts.

Usage

```
cusum_failure(xi, p0, p1, by = NULL, alpha = 0.05, beta = 0.05)
```

Arguments

xi	An integer. The dichotomous outcome variable (1 = Failure, 0 = Success) for the i-th procedure.
p0	A double. The acceptable event rate.
p1	A double. The unacceptable event rate.
by	A factor. Optional variable to stratify procedures by.

alpha	A double. The Type I Error rate. Probability of rejecting the null hypothesis when ‘p0‘ is the true rate.
beta	A double. The Type II Error rate. Probability of failing to reject null hypothesis when it is false.

Value

An object of class `data.frame`.

References

Rogers, C. A., Reeves, B. C., Caputo, M., Ganesh, J. S., Bonser, R. S., & Angelini, G. D. (2004). Control chart methods for monitoring cardiac surgical performance and their interpretation. The Journal of Thoracic and Cardiovascular Surgery, 128(6), 811-819.

Examples

```
library(purrr)
library(ggplot2)

# Data
df <- data.frame(
  xi = simplify(
    map(
      c(.1,.08,.05,.1,.13,.14,.14,.09,.25),
      ~ rbinom(50,1,.x))),
   p0 = simplify(
    map(
      c(.1,.1,.1,.1,.1,.1,.1,.15,.2),
      ~ rnorm(50,.x,.03))),
   by = rep(
    factor(paste('Subject', c('A','B','C'))),
    times = c(150,150,150))
 )

# Overall event rate
p0 <- sum(df$xi) / nrow(df)

# Create CUSUM plot
cusum_failure(
 xi = df$xi,
 p0 = p0,
 p1 = p0 * 1.5,
 by = df$by
) |>
ggplot(aes(y = cusum, x = i)) +
  geom_step() +
  geom_line(mapping = aes(y = l0), linetype = 2) +
  geom_line(mapping = aes(y = l1), linetype = 2) +
  ylab("Cumulative Failures") +
  xlab("Case Number") +
  facet_wrap(~ by) +
```

```
  theme_bw()
```

`cusum_loglike` *Cumulative Sum of Log-Likelihood Ratio*

Description

Calculates the cumulative log likelihood ratio of failure for a series of procedures which can be used to create CUSUM charts.

Usage

```
cusum_loglike(xi, p0, p1, by = NULL, alpha = 0.05, beta = 0.05)
```

Arguments

`xi`	An integer. The dichotomous outcome variable (1 = Failure, 0 = Success) for the i-th procedure.
`p0`	A double. The acceptable event rate.
`p1`	A double. The unacceptable event rate.
`by`	A factor. Optional variable to stratify procedures by.
`alpha`	A double. The Type I Error rate. Probability of rejecting the null hypothesis when 'p0' is true.
`beta`	A double. The Type II Error rate. Probability of failing to reject null hypothesis when it is false.

Value

An object of class `data.frame`.

References

Rogers, C. A., Reeves, B. C., Caputo, M., Ganesh, J. S., Bonser, R. S., & Angelini, G. D. (2004). Control chart methods for monitoring cardiac surgical performance and their interpretation. The Journal of Thoracic and Cardiovascular Surgery, 128(6), 811-819.

Examples

```
library(purrr)
library(ggplot2)

# Data
df <- data.frame(
  xi = simplify(
    map(
      c(.1,.08,.05,.1,.13,.14,.14,.09,.25),
      ~ rbinom(50,1,.x))),
```

```
   p0 = simplify(
    map(
      c(.1,.1,.1,.1,.1,.1,.1,.15,.2),
      ~ rnorm(50,.x,.03))),
   by = rep(
    factor(paste('Subject', c('A','B','C'))),
    times = c(150,150,150))
 )

# Overall event rate
p0 <- sum(df$xi) / nrow(df)

# Create CUSUM plot
cusum_loglike(
  xi = df$xi,
  p0 = p0,
  p1 = p0 * 1.5,
  by = df$by
) |>
ggplot(aes(y = cusum, x = i)) +
  geom_step() +
  geom_hline(aes(yintercept = h0), linetype = 2) +
  geom_hline(aes(yintercept = h1), linetype = 2) +
  ylab("Cumulative Log-likelihood Ratio") +
  xlab("Case Number") +
  facet_wrap(~ by) +
  theme_bw()
```

`cusum_ome` *Cumulative Sum of Observed Minus Expected Outcome*

Description

Calculates the cumulative observed-minus-expected failure for a series of procedures which can be used to create CUSUM charts.

Usage

```
cusum_ome(xi, p0, by = NULL)
```

Arguments

`xi`	An integer. The dichotomous outcome variable (1 = Failure, 0 = Success) for the i-th procedure.
`p0`	A double. The acceptable event rate.
`by`	A factor. Optional variable to stratify procedures by.

Value

An object of class `data.frame`.

References

Rogers, C. A., Reeves, B. C., Caputo, M., Ganesh, J. S., Bonser, R. S., & Angelini, G. D. (2004). Control chart methods for monitoring cardiac surgical performance and their interpretation. The Journal of Thoracic and Cardiovascular Surgery, 128(6), 811-819.

Examples

```
library(purrr)
library(ggplot2)

# Data
df <- data.frame(
  xi = simplify(
    map(
      c(.1,.08,.05,.1,.13,.14,.14,.09,.25),
      ~ rbinom(50,1,.x))),
   p0 = simplify(
    map(
      c(.1,.1,.1,.1,.1,.1,.1,.15,.2),
      ~ rnorm(50,.x,.03))),
   by = rep(
    factor(paste('Subject', c('A','B','C'))),
    times = c(150,150,150))
 )

# Create CUSUM plot
cusum_ome(
 xi = df$xi,
 p0 = df$p0,
 by = df$by
) |>
ggplot(aes(x = i, y = cusum)) +
geom_hline(yintercept = 0, linetype = 6, linewidth = 0.5) +
  geom_step() +
  ylab("Cumulative Observed Minus Expected Failures") +
  xlab("Case Number") +
  facet_wrap(~ by) +
  theme_bw()
```

`cusum_sprt` *Risk-adjusted Sequential Probability Ratio Test (SPRT)*

Description

Calculates the risk-adjusted sequential probability ratio test for a series of procedures which can be used to create CUSUM charts.

Usage

```
cusum_sprt(xi, p0, OR, by = NULL, alpha = 0.05, beta = 0.05)
```

Arguments

xi	An integer. The dichotomous outcome variable (1 = Failure, 0 = Success) for the i-th procedure.
p0	A double. The individual acceptable event rate for each individual procedure (adjusted).
OR	A double. An odds-ratio reflecting the increase in relative risk of failure.
by	A factor. Optional variable to stratify procedures by.
alpha	A double. The Type I Error rate. Probability of rejecting the null hypothesis when ‘p0‘ is true.
beta	A double. The Type II Error rate. Probability of failing to reject null hypothesis when it is false.

Value

An object of class `data.frame`.

References

Rogers, C. A., Reeves, B. C., Caputo, M., Ganesh, J. S., Bonser, R. S., & Angelini, G. D. (2004). Control chart methods for monitoring cardiac surgical performance and their interpretation. The Journal of Thoracic and Cardiovascular Surgery, 128(6), 811-819.

Examples

```
library(purrr)
library(ggplot2)

# Data
df <- data.frame(
  xi = simplify(
    map(
      c(.1,.08,.05,.1,.13,.14,.14,.09,.25),
      ~ rbinom(50,1,.x))),
   p0 = simplify(
    map(
      c(.1,.1,.1,.1,.1,.1,.1,.15,.2),
      ~ rnorm(50,.x,.03))),
   by = rep(
    factor(paste('Subject', c('A','B','C'))),
    times = c(150,150,150))
 )

# Create CUSUM plot
cusum_sprt(
  xi = df$xi,
  p0 = df$p0,
  OR = 1.5,
  by = df$by
) |>
```

```
ggplot(aes(y = cusum, x = i)) +
  geom_step() +
  geom_hline(aes(yintercept = h0), linetype = 2) +
  geom_hline(aes(yintercept = h1), linetype = 2) +
  ylab("Cumulative Log-likelihood Ratio") +
  xlab("Case Number") +
  facet_wrap(~ by) +
  theme_bw()
```

`paste` *Concatenate strings*

Description

An augmented version of `base::paste()` with options to manage 'NA' values.

Usage

```
paste(..., sep = " ", collapse = NULL, na.rm = FALSE)

paste0(..., collapse = NULL, na.rm = FALSE)
```

Arguments

`...`	R objects to be converted to character vectors.
`sep`	A character. A string to separate the terms.
`collapse`	A character. An string to separate the results.
`na.rm`	A logical. Whether to remove NA values from 'x'.

Value

Character vector of concatenated values.

See Also

`paste`

Examples

```
# Base paste() NA handling behavior
paste(
  'The', c('red', NA_character_, 'orange'), 'fox jumped', NA_character_, 'over the fence.',
  collapse = ' '
)

# Removal of NA values
paste(
  'The', c('red', NA_character_, 'orange'), 'fox jumped', NA_character_, 'over the fence.',
```

```
  collapse = ' ',
  na.rm = TRUE
)
```

`paste_efs` *Paste event-free survival*

Description

Creates a formatted event-free-survival from a survfit object and a specified time point.

Usage

```
paste_efs(x, times, percent.sign = TRUE, digits = 1)
```

Arguments

`x`	A `survfit` object. The survival model.
`times`	A numeric. Indicates time-points of interest. Units are whatever was used to create the survival fit.
`percent.sign`	A logical. Indicates percent sign should be printed for frequencies.
`digits`	Integer. Number of digits to round to.

Value

A named character vector of event-free survival(s).

Examples

```
library(survival)

fit <- survfit(Surv(time, status) ~ 1, data = diabetic)
paste_efs(fit, c(1, 3, 5))
```

`paste_freq` *Paste frequency*

Description

Creates a formatted frequency from count(able) data. Automatically tallies non-numeric data types (nrow or length) and supports vectorized data methods.

Usage

```
paste_freq(x, y, na.rm = TRUE, percent.sign = TRUE, digits = 1)
```

Arguments

x	A data.frame, numeric, or non-numeric. The numerator.
y	A data.frame, numeric, or non-numeric. The denominator. A single denominator may be used for multiple numerators or one denominator for each numerator.
na.rm	A logical. Whether to ignore NA's when tallying non-numeric data.
percent.sign	A logical. Indicates percent sign should be printed with frequencies.
digits	An integer. Number of digits to round to.

Value

A character vector of count(s) with frequencies.

Examples

```
# Numeric
paste_freq(20, 100)

# data.frame
df <- data.frame(x = c(1:100), y = TRUE)
paste_freq(df[1:20,], df)

# Mixed data types
paste_freq(20, df)

# Single denominator for multiple numerators
paste_freq(c(10,20,30), 100)
```

---

paste_mean *Paste mean*

Description

Creates a formatted mean with standard deviation from numeric data.

Usage

```
paste_mean(x, less.than.one = FALSE, digits = 1)
```

Arguments

x	A numeric. Data to summarize.
less.than.one	A logical. Indicates a mean that rounds to 0 should be printed as <1.
digits	An integer. Number of digits to round to.

Value

A character vector of the mean(s) with standard deviation(s).

Examples

```
paste_mean(mtcars$mpg)
```

`paste_median` *Paste median*

Description

Creates a formatted median with inter-quartile range from numeric data.

Usage

```
paste_median(x, less.than.one = FALSE, digits = 1)
```

Arguments

`x`	A numeric. Data to summarize.
`less.than.one`	A logical. Indicates a median that rounds to 0 should be printed as <1.
`digits`	An integer. Number of digits to round to.

Value

A character vector of the median(s) with interquartile range(s).

Examples

```
paste_median(mtcars$mpg)
```

`paste_pval` *Paste p-value*

Description

Creates a human-readable p.value using sensible defaults for ‘format.pval()‘.

Usage

```
paste_pval(x, digits = 1, p.digits = 4)
```

Arguments

`x`	A numeric. P-value to format.
`digits`	A numeric. Number of significant digits to round to.
`p.digits`	A numeric. Minimum number of digits to right of the decimal point.

Examples

```
paste_pval(0.061126e-10)
```

`test_hypothesis` *Test the null hypothesis*

Description

Tests the null hypothesis that there is no difference between grouped data.

Usage

```
test_hypothesis(
  x,
  y,
  test,
  digits,
  p.digits,
  simulate.p.value,
  B,
  workspace,
  ...
)

## S3 method for class 'numeric'
test_hypothesis(
  x,
  y,
  test = c("anova", "kruskal", "wilcoxon"),
  digits = 1,
  p.digits,
  ...
)

## S3 method for class 'factor'
test_hypothesis(
  x,
  y,
  test = c("chisq", "fisher"),
  digits = 1,
  p.digits,
  simulate.p.value = FALSE,
  B = 2000,
  workspace = 2e+07,
  ...
)

## S3 method for class 'logical'
test_hypothesis(
  x,
```

```
  y,
  test = c("chisq", "fisher"),
  digits = 1,
  p.digits,
  simulate.p.value = FALSE,
  B = 2000,
  workspace = 2e+07,
  ...
)
```

Arguments

x	A numeric, factor, or logical. Observations.
y	A factor or logical. Categorical "by" grouping variable.
test	A character. Name of the statistical test to use. See note.
digits	An integer. Number of digits to round to.
p.digits	An integer. The number of p-value digits to the right of the decimal point. Note that p-values are still rounded using 'digits'.
simulate.p.value	A logical. Whether p-values in nominal variable testing should be computed with Monte Carlo simulation.
B	An integer. Number of replicates to use in Monte Carlo simulation for nominal testing.
workspace	An integer. Size of the workspace used for the Fisher's Exact Test network algorithm.
...	Additional arguments passed to the appropriate S3 method.

Value

A list containing the statistical test performed, test statistic, and p-value.

Note

Statistical testing used is dependent on type of 'x' data. Supported testing for numeric data includes ANOVA ('anova'), Kruskal-Wallis ('kruskal'), and Wilcoxon Rank Sum ('wilcoxon') tests. For categorical data, supported testings includes Pearson's Chi-squared ('chisq') and Fisher's Exact Test ('fisher').

Examples

```
strata <- as.factor(mtcars$cyl)

# Numeric data
test_hypothesis(mtcars$mpg, strata)

# Logical data
test_hypothesis(as.logical(mtcars$vs), strata)
```

```
# Factor data
test_hypothesis(as.factor(mtcars$carb), strata)
```

Index