Package 'traumar'

February 21, 2025

Type Package

Title Calculate Metrics for Trauma System Performance

Version 1.0.0

Description Hospitals, hospital systems, and even trauma systems that provide care to injured patients may not be aware of robust metrics that can help gauge the efficacy of their programs in saving the lives
of injured patients. 'traumar' provides robust functions driven by the academic literature to automate the calculation of relevant metrics to individuals desiring to measure the performance of their
trauma center or even a trauma system. 'traumar' also provides some helper functions for the data analysis
journey. Users can refer to the following publications for descriptions of the methods used in 'traumar'.
TRISS methodology, including probability of survival, and the W, M, and Z Scores - Flora (1978) <doi:10.1097/00005373-197810000-00003>, Boyd et al. (1987, PMID:3106646), Llullaku et al. (2009) <doi:10.1186/1749-7922-4-2>, Singh et al. (2011) <doi:10.4103/0974-2700.86626>, Baker et al. (1974, PMID:4814394),
and Champion et al. (1989) <doi:10.1097/00005373-198905000-00017>. For the Relative Mortality Metric,
see Napoli et al. (2017) <doi:10.1080/24725579.2017.1325948>, Schroeder et al. (2019) <doi:10.1080/10903127.2018.1489021>, and Kassar et al. (2016) <doi:10.1177/00031348221093563>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.1)

Suggests broom, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports cli (>= 3.6.3), dplyr (>= 1.1.4), ggplot2 (>= 3.5.1), rlang (>= 1.1.5), stats, tibble (>= 3.2.1), tidyr (>= 1.3.1), tidyselect (>= 1.2.1), utils, lubridate (>= 1.9.4), lifecycle (>= 1.0.4), infer (>= 1.0.7), purrr (>= 1.0.2)

URL https://bemts-hhs.github.io/traumar/,
https://github.com/bemts-hhs/traumar

BugReports https://github.com/bemts-hhs/traumar/issues

NeedsCompilation no

Author Nicolas Foss [aut, cre],
Iowa Department of Health and Human Services [cph]

Maintainer Nicolas Foss <nicolas.foss@hhs.iowa.gov>

Repository CRAN

Date/Publication 2025-02-21 11:50:07 UTC

Contents

impute 2
nonlinear_bins 3
normalize 5
pretty_number 6
pretty_percent 7
probability_of_survival 8
rmm 9
rm_bin_summary 12
season 14
small_count_label 15
stat_sig 16
theme_cleaner 17
trauma_case_mix 19
trauma_performance 20
weekend 22
%not_in% 23

Index **25**

`impute` *Impute Numeric Column Values*

Description

Cleans numeric columns by handling extreme values or imputing missing values. The function supports two main focuses: handling skewed distributions or imputing missing data.

Usage

```
impute(
  x,
  focus = c("skew", "missing"),
  method = c("winsorize", "iqr", "mean", "median"),
  percentile = NULL
)
```

Arguments

x	A numeric vector to be cleaned.
focus	A character string indicating the focus. Options are: • "skew": Handle extreme values using percentile or IQR methods (default). • "missing": Impute missing values.
method	A character string specifying the method: • For focus = "skew": – "winsorize": Replace values outside specified percentiles (default). – "iqr": Use IQR to limit extreme values. • For focus = "missing": – "mean": Replace missing values with the mean. – "median": Replace missing values with the median.
percentile	A numeric value (percentile > 0) for winsorization. If not provided, defaults to 0.01 and 0.99.

Value

A numeric vector with cleaned or imputed values.

Examples

```
x <- c(1, 2, 3, 100, 200, NA)
# Winsorize to 1% and 99%
impute(x, focus = "skew", method = "winsorize")

# Replace missing values with the mean
impute(x, focus = "missing", method = "mean")
```

---

nonlinear_bins *Create Nonlinear Probability of Survival Bins*

Description

This function generates nonlinear bins for probability of survival data based on specified thresholds and divisors as specified in Napoli et al. (2017), Schroeder et al. (2019), and Kassar et al. (2016). This function calculates bin statistics, including mean, standard deviation, total alive, total dead, count, and percentage for each bin.

Usage

```
nonlinear_bins(
  data,
  Ps_col,
  outcome_col,
  divisor1 = 5,
  divisor2 = 5,
  threshold_1 = 0.9,
  threshold_2 = 0.99
)
```

Arguments

`data`	A `data.frame` or `tibble` containing the probability of survival data for a set of patients.
`Ps_col`	The column in `data` containing the probability of survival values for a set of patients.
`outcome_col`	The name of the column containing the outcome data. It should be binary, with values indicating patient survival. A value of `1` should represent "alive" (survived), while `0` should represent "dead" (did not survive). Ensure the column contains only these two possible values.
`divisor1`	A parameter to control the width of the probability of survival range bins. Affects the creation of step sizes for the beginning of each bin range. Defaults to `5`.
`divisor2`	A parameter to control the width of the probability of survival range bins. Affects the creation of step sizes for the beginning of each bin range. Defaults to `5`.
`threshold_1`	A parameter to decide where data indices will begin to create step sizes. Defaults to `0.9`.
`threshold_2`	A parameter to decide where data indices will end to create step sizes. Defaults to `0.99`.

Value

A list with `intervals` and `bin_stats` objects:

- `intervals`: A vector of start and end-points for the probability of survival bin ranges.
- `bin_stats`: A `tibble` with columns `bin_number`, `bin_start`, `bin_end`, `mean`, `sd`, `alive`, `dead`, `count`, and `percent`.

Author(s)

Nicolas Foss, Ed.D, MS, original paper and code in MATLAB by Nicholas J. Napoli, Ph.D., MS

Examples

```
# Generate example data with high negative skewness
set.seed(123)

# Parameters
n_patients <- 10000  # Total number of patients

# Skewed towards higher values
Ps <- plogis(rnorm(n_patients, mean = 2, sd = 1.5))

# Simulate survival outcomes based on Ps
survival_outcomes <- rbinom(n_patients,
                            size = 1,
                            prob = Ps
                            )

# Create data frame
data <- data.frame(Ps = Ps, survival = survival_outcomes) |>
dplyr::mutate(death = dplyr::if_else(survival == 1, 0, 1))

# Apply the nonlinear_bins function
results <- nonlinear_bins(data = data,
                          Ps_col = Ps,
                          outcome_col = survival,
                          divisor1 = 5,
                          divisor2 = 5,
                          threshold_1 = 0.9,
                          threshold_2 = 0.99)

# View results
results$intervals
results$bin_stats
```

`normalize` *Normalize a Numeric Vector*

Description

This function normalizes a numeric or integer vector using one of two methods: min-max normalization (scales data to the range (0, 1)) or z-score normalization (centers data around 0 with a standard deviation of 1).

Usage

```
normalize(x, method = c("min_max", "z_score"))
```

Arguments

x	A numeric or integer vector to be normalized.
method	A character string specifying the normalization method. Options are "min_max" for min-max normalization or "z_score" for z-score normalization. If no method is provided, the default is "min_max".

Value

A numeric vector of the same length as x, containing the normalized values.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Example data
data <- c(10, 20, 30, 40, 50, NA)

# Min-max normalization
normalize(data, method = "min_max")

# Z-score normalization
normalize(data, method = "z_score")

# Default behavior (min-max normalization)
normalize(data)
```

---

pretty_number *Convert Numbers into Readable Abbreviated Formats*

Description

This function converts large numeric values into readable abbreviated formats (e.g., 1,000 becomes "1k") with options for rounding, decimal precision, and a custom prefix. It supports numbers up to the decillion range.

Usage

```
pretty_number(x, n_decimal = 2, prefix = NULL, truncate = FALSE)
```

Arguments

x	A numeric value or vector to be converted into a readable format.
n_decimal	An integer specifying the number of decimal places to include in the output. Defaults to 2.
prefix	An optional character string to prepend to the formatted number (e.g., "$"). Defaults to NULL.
truncate	A logical value indicating whether to truncate the numbers before formatting. Defaults to FALSE.

Value

A character vector with the numbers formatted as abbreviated strings. If prefix is provided, it prepends the formatted numbers.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Basic usage
pretty_number(1234)               # "1.23k"
pretty_number(1234567)            # "1.23m"
pretty_number(1234567890)         # "1.23b"

# Adjusting decimal places
pretty_number(1234, n_decimal = 1) # "1.2k"

# Adding a prefix
pretty_number(1234, prefix = "$")  # "$1.23k"

# Without rounding
pretty_number(1250, truncate = TRUE) # "1.2k"
```

---

pretty_percent *Format Numeric Variables as Percentages*

Description

This function formats numeric variables as percentages with a specified number of decimal places. It refines the output by removing unnecessary trailing zeros after the decimal point and ensures the percentage sign is correctly applied without extraneous characters, resulting in a polished, human-readable percentage representation.

Usage

```
pretty_percent(variable, n_decimal = 1)
```

Arguments

variable	A numeric vector representing proportions to format as percentages. The values are on a scale from 0 to 1.
n_decimal	A numeric value specifying the number of decimal places. Defaults to 1.

Value

A character vector containing the formatted percentages.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Example usage:
pretty_percent(0.12345)  # Default decimal places
pretty_percent(0.12345, n_decimal = 2)  # Two decimal places
pretty_percent(c(0.1, 0.25, 0.3333), n_decimal = 1)  # Vector input
```

---

probability_of_survival

Calculate Probability of Survival Using TRISS Method

Description

This function calculates the probability of survival (Ps) for trauma patients based on the Trauma and Injury Severity Score (TRISS) methodology. TRISS combines physiological and anatomical data to predict survival likelihood using a logistic regression model. The function incorporates trauma type, patient age, Revised Trauma Score (RTS), and Injury Severity Score (ISS) into the calculation. Probability of survival is expressed as a percentage.

Usage

```
probability_of_survival(trauma_type, age, rts, iss)
```

Arguments

trauma_type	Character vector indicating the type of trauma ("Blunt" or "Penetrating"). Different methods exist for calculating probability of survival for burn patients, and so these records are excluded here.
age	Numeric vector indicating the patient's age in years.
rts	Numeric vector indicating the patient's Revised Trauma Score (RTS).
iss	Numeric vector indicating the patient's Injury Severity Score (ISS).

Value

Numeric vector of probabilities of survival (Ps) expressed as percentages on a scale from 0 to 1.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Example usage:
trauma_data <- data.frame(
  Trauma_Type = c("Blunt", "Penetrating"),
  Patient_Age_Years = c(30, 60),
  RTS = c(7.84, 6.90),
  ISS = c(10, 25)
)

result <- trauma_data |>
  dplyr::mutate(Ps = probability_of_survival(
    trauma_type = Trauma_Type,
    age = Patient_Age_Years,
    rts = RTS,
    iss = ISS
  ))
```

---

rmm *Relative Mortality Metric (RMM) Calculation*

Description

Calculates the Relative Mortality Metric (RMM) from Napoli et al. (2017) based on patient survival probabilities (Ps) and actual outcomes. The function groups patients into bins based on their survival probability scores (Ps) and computes a weighted mortality metric along with confidence intervals. For more information on the methods used in this function, see as well Schroeder et al. (2019), and Kassar et al. (2016).

The Relative Mortality Metric (RMM) quantifies the performance of a center in comparison to the anticipated mortality based on the TRISS national benchmark. The RMM measures the difference between observed and expected mortality, with a range from -1 to 1.

- An RMM of 0 indicates that the observed mortality aligns with the expected national benchmark across all acuity levels.
- An RMM greater than 0 indicates better-than-expected performance, where the center is outperforming the national benchmark.
- An RMM less than 0 indicates under-performance, where the center's observed mortality is higher than the expected benchmark.

This metric helps assess how a center's mortality compares to the national standards, guiding quality improvement efforts. `rmm()` utilizes bootstrap sampling to calculate the confidence intervals via the standard error method.

Due to the use of bootstrap sampling within the function, users should set the random number seed before running `rmm()`.

Usage

```
rmm(
  data,
  Ps_col,
  outcome_col,
  n_samples = 1000,
  Divisor1 = 5,
  Divisor2 = 5,
  Threshold_1 = 0.9,
  Threshold_2 = 0.99,
  pivot = FALSE
)
```

Arguments

`data`	A data frame or tibble containing the data.
`Ps_col`	The name of the column containing the survival probabilities (Ps). Should be numeric (values between 0 and 100).
`outcome_col`	The name of the column containing the outcome data. It should be binary, with values indicating patient survival. A value of `1` should represent "alive" (survived), while `0` should represent "dead" (did not survive). Ensure the column contains only these two possible values.
`n_samples`	A numeric value indicating the number of bootstrap samples to take from the data source.
`Divisor1`	A divisor used for binning the survival probabilities (default is 5).
`Divisor2`	A second divisor used for binning the survival probabilities (default is 5).
`Threshold_1`	The first threshold for dividing the survival probabilities (default is 0.9).
`Threshold_2`	The second threshold for dividing the survival probabilities (default is 0.99).
`pivot`	A logical indicating whether to return the results in a long format (pivot = TRUE) or wide format (pivot = FALSE, default).

Value

A tibble containing the Relative Mortality Metric (RMM) and related statistics:

- `population_RMM_LL`: The lower bound of the 95% confidence interval for the population RMM.
- `population_RMM`: The final calculated Relative Mortality Metric for the population existing in `data`.

- `population_RMM_UL`: The upper bound of the 95% confidence interval for the population RMM.
- `population_CI`: The confidence interval width for the population RMM.
- `bootstrap_RMM_LL`: The lower bound of the 95% confidence interval for the bootstrap RMM.
- `bootstrap_RMM`: The average RMM value calculated for the bootstrap sample.
- `bootstrap_RMM_UL`: The upper bound of the 95% confidence interval for the bootstrap RMM.
- `bootstrap_CI`: The width of the 95% confidence interval for the bootstrap RMM.
- If `pivot = TRUE`, the results will be in long format with two columns: `stat` and `value`, where each row corresponds to one of the calculated statistics.
- If `pivot = FALSE` (default), the results will be returned in wide format, with each statistic as a separate column.

Author(s)

Nicolas Foss, Ed.D, MS, original paper and code in MATLAB by Nicholas J. Napoli, Ph.D., MS

Examples

```
# Generate example data with high negative skewness
set.seed(10232015)

# Parameters
n_patients <- 1000  # Total number of patients

# Skewed towards higher values
Ps <- plogis(rnorm(n_patients, mean = 2, sd = 1.5))

# Simulate survival outcomes based on Ps
survival_outcomes <- rbinom(n_patients,
                            size = 1,
                            prob = Ps
                            )

# Create data frame
data <- data.frame(Ps = Ps, survival = survival_outcomes) |>
dplyr::mutate(death = dplyr::if_else(survival == 1, 0, 1))

# Example usage of the `rmm` function
rmm(data = data, Ps_col = Ps,
    outcome_col = survival,
    Divisor1 = 5,
    Divisor2 = 5,
    n_samples = 5
    )

# pivot!
rmm(data = data, Ps_col = Ps,
    outcome_col = survival,
    Divisor1 = 5,
```

```
Divisor2 = 5,
n_samples = 5,
pivot = TRUE
)
```

`rm_bin_summary` *Bin-Level Summary for Relative Mortality Metric (RMM)*

Description

Calculates a bin-level summary for the Relative Mortality Metric (RMM) from Napoli et al. (2017) by grouping data into bins based on survival probabilities (Ps) and summarizing outcomes within each bin. This function returns statistics such as total alive, total dead, estimated mortality, anticipated mortality, and confidence intervals for each bin. For more information on the methods used in this function, see as well Schroeder et al. (2019), and Kassar et al. (2016).

The Relative Mortality Metric (RMM) quantifies the performance of a center in comparison to the anticipated mortality based on the TRISS national benchmark. The RMM measures the difference between observed and expected mortality, with a range from -1 to 1.

- An RMM of 0 indicates that the observed mortality aligns with the expected national benchmark across all acuity levels.
- An RMM greater than 0 indicates better-than-expected performance, where the center is outperforming the national benchmark.
- An RMM less than 0 indicates under-performance, where the center's observed mortality is higher than the expected benchmark.

This metric helps assess how a center's mortality compares to the national standards, guiding quality improvement efforts.`rm_bin_summary()` utilizes bootstrap sampling to calculate the confidence intervals via the standard error method.

Due to the use of bootstrap sampling within the function, users should set the random number seed before running `rm_bin_summary()`.

Usage

```
rm_bin_summary(
  data,
  Ps_col,
  outcome_col,
  n_samples = 1000,
  Divisor1 = 5,
  Divisor2 = 5,
  Threshold_1 = 0.9,
  Threshold_2 = 0.99
)
```

Arguments

data	A data frame or tibble containing the data.
Ps_col	The name of the column containing the survival probabilities (Ps). Should be numeric (values between 0 and 100).
outcome_col	The name of the column containing the outcome data. It should be binary, with values indicating patient survival. A value of 1 should represent "alive" (survived), while 0 should represent "dead" (did not survive). Ensure the column contains only these two possible values.
n_samples	A numeric value indicating the number of bootstrap samples to take from the data source.
Divisor1	A divisor used for binning the survival probabilities (default is 5).
Divisor2	A second divisor used for binning the survival probabilities (default is 5).
Threshold_1	The first threshold for dividing the survival probabilities (default is 0.9).
Threshold_2	The second threshold for dividing the survival probabilities (default is 0.99).

Value

A tibble containing bin-level statistics including:

- bin_number: The bin to which each record was assigned.
- TA_b: Total alive in each bin (number of patients who survived).
- TD_b: Total dead in each bin (number of patients who did not survive).
- N_b: Total number of patients in each bin.
- EM_b: Estimated mortality rate for each bin (TD_b / (TA_b + TD_b)).
- AntiS_b: The anticipated survival rate for each bin.
- AntiM_b: The anticipated mortality rate for each bin.
- bin_start: The lower bound of the survival probability range for each bin.
- bin_end: The upper bound of the survival probability range for each bin.
- midpoint: The midpoint of the bin range (calculated as (bin_start + bin_end) / 2).
- R_b: The width of each bin (bin_end - bin_start).
- population_RMM_LL: The lower bound of the 95% confidence interval for the population RMM.
- population_RMM: The final calculated Relative Mortality Metric for the population existing in data.
- population_RMM_UL: The upper bound of the 95% confidence interval for the population RMM.
- population_CI: The confidence interval width for the population RMM.
- bootstrap_RMM_LL: The lower bound of the 95% confidence interval for the bootstrap RMM.
- bootstrap_RMM: The average RMM value calculated for the bootstrap sample.
- bootstrap_RMM_UL: The upper bound of the 95% confidence interval for the bootstrap RMM.
- bootstrap_CI: The width of the 95% confidence interval for the bootstrap RMM.

Author(s)

Nicolas Foss, Ed.D, MS, original paper and code in MATLAB by Nicholas J. Napoli, Ph.D., MS

Examples

```
# Generate example data with high negative skewness
set.seed(10232015)

# Parameters
n_patients <- 10000  # Total number of patients

Ps <- plogis(rnorm(n_patients, mean = 2,
                   sd = 1.5)
                   )  # Skewed towards higher values

# Simulate survival outcomes based on Ps
survival_outcomes <- rbinom(n_patients,
                            size = 1,
                            prob = Ps
                            )

# Create data frame
data <- data.frame(Ps = Ps, survival = survival_outcomes) |>
dplyr::mutate(death = dplyr::if_else(survival == 1, 0, 1))

# Example usage of the `rm_bin_summary` function
rm_bin_summary(data = data, Ps_col = Ps,
               outcome_col = survival,
               n_samples = 5
               )
```

---

season *Get Season Based on a Date*

Description

This function determines the season (Winter, Spring, Summer, or Fall) based on an input date.

Usage

```
season(input_date)
```

Arguments

input_date	A Date or POSIXct object representing the date to determine the season for. The input must be of class `Date` or `POSIXct`.

Details

The seasons are assigned based on geographic regions similar to how seasons occur in the United States.

The seasons are determined using the month of the year and the traditional meteorological definition of seasons (Winter: December, January, February; Spring: March, April, May; Summer: June, July, August; Fall: September, October, November).

Value

A factor indicating the season corresponding to the input date. The factor levels are:

- "Winter" for December, January, and February.
- "Spring" for March, April, and May.
- "Summer" for June, July, and August.
- "Fall" for September, October, and November.
- "Undetermined" if the input is not a valid Date or POSIXct object or if the month is missing.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Example usage of the season function
season(as.Date("2025-01-15"))
season(as.POSIXct("2025-07-01 12:00:00"))
```

---

small_count_label *Label Small Counts Based on a Cutoff*

Description

This function labels values in a vector as a replacement string if they are smaller than a specified cutoff. The input can be numeric, and the function will return either a modified version of the input vector with small values replaced by a given label, or it will keep the original values otherwise.

Usage

```
small_count_label(var, cutoff, replacement)
```

Arguments

var	A numeric vector. This represents the variable to be checked against the cutoff.
cutoff	A numeric value representing the threshold. Values in var smaller than this value will be replaced.
replacement	A string or a numeric value. If the value in var is smaller than the cutoff, this value will replace it. If a string is provided, it will replace the numeric values with the string. If a numeric value is provided, the replacement will also be numeric.

Value

A vector with values from var. Values smaller than the cutoff will be replaced by the replacement. If replacement is a string, the return type will be character, otherwise, it will remain numeric.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Example usage of the small_count_label function
small_count_label(c(1, 5, 10), 5, "Below Cutoff")
small_count_label(c(1, 5, 10), 5, 0)
```

---

stat_sig *Assign Significance Codes Based on P-Values*

Description

This function assigns significance codes to a p-value vector based on commonly accepted significance thresholds. The significance codes are:

- "***" for p-values <= 0.001
- "**" for p-values <= 0.01 and > 0.001
- "*" for p-values <= 0.05 and > 0.01
- "." for p-values <= 0.1 and > 0.05
- "<>" for p-values > 0.1

Usage

```
stat_sig(p_val_data)
```

Arguments

p_val_data	A numeric vector representing the p-values to be categorized. The vector should contain p-values between 0 and 1.

Value

A character vector with the assigned significance codes for each p-value.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Example usage of the stat_sig function
data <- data.frame(p_value = c(0.001, 0.03, 0.12, 0.05, 0.07))

data |>
  dplyr::mutate(significance = stat_sig(p_val_data = p_value))
```

---

theme_cleaner *Customizable Minimalistic ggplot2 Theme*

Description

A flexible and customizable theme function for creating polished and minimalistic plots using ggplot2. The theme_cleaner function provides various options to control the appearance of plot elements, including font styles, sizes, colors, axis lines, grid lines, legend, title, subtitle, captions, and facet appearance. The theme is highly customizable, allowing for the creation of visually appealing and clean plots.

Usage

```
theme_cleaner(
  base_size = 12,
  base_family = "sans",
  base_color = "#70C8B8",
  base_color_title = "#03617A",
  title_text_size = ceiling(base_size * 1.1),
  subtitle_text_size = ceiling(base_size * 1.05),
  caption_color = "#19405B",
  legend_position = "top",
  vjust_title = 0,
  vjust_subtitle = 0,
  hjust_title = 0,
  hjust_subtitle = 0,
  axis_lines = FALSE,
  facets = FALSE,
  facet_text_size = base_size,
  draw_panel_border = FALSE,
  ...
)
```

Arguments

base_size	Numeric. Default font size for plot elements. Defaults to 12.
base_family	Character. Font family used for text in the plot. Defaults to "Work Sans".
base_color	Character. Hex color code for primary plot elements (e.g., axis text, legend text). Defaults to "#70C8B8".
base_color_title	Character. Hex color code for plot title and legend title text. Defaults to "#03617A".
title_text_size	Numeric. Font size for plot title text. Defaults to base_size * 1.1.
subtitle_text_size	Numeric. Font size for plot subtitle text. Defaults to base_size * 1.05.
caption_color	Character. Hex color code for plot caption text. Defaults to "#19405B".
legend_position	Character. Legend position on the plot. Accepts "top", "bottom", "left", or "right". Defaults to "top".
vjust_title	Numeric. Vertical justification of the plot title. Defaults to 0.
vjust_subtitle	Numeric. Vertical justification of the plot subtitle. Defaults to 0.
hjust_title	Numeric. Horizontal justification of the plot title. Defaults to 0.
hjust_subtitle	Numeric. Horizontal justification of the plot subtitle. Defaults to 0.
axis_lines	Logical. If TRUE, axis lines are drawn in base_color; otherwise, they are hidden. Defaults to FALSE.
facets	Logical. If TRUE, additional formatting for facet plots is applied. Defaults to FALSE.
facet_text_size	Numeric. If facets = TRUE, size formatting for facet text (strip.text) is applied. Defaults to base_size.
draw_panel_border	Logical. If TRUE, a border is drawn around panels in facet plots. Defaults to FALSE.
...	Additional arguments passed to ggplot2::theme for further customization.

Details

The function customizes common plot elements like axis text, titles, subtitles, captions, legend, and facet labels. It is designed to work with ggplot2 plots, providing a clean and professional look with minimal styling. You can adjust various aesthetic features such as font size, color, and legend position for maximum control over the appearance.

Value

A ggplot2 theme object that can be applied to plots.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Create a ggplot2 plot with the theme_cleaner theme
library(ggplot2)
ggplot(mtcars, aes(x = mpg, y = wt)) +
  geom_point() +
  theme_cleaner(base_size = 14, title_text_size = 16, legend_position = "bottom")

# Customize facet plots with theme_cleaner
ggplot(mtcars, aes(x = mpg, y = wt)) +
  geom_point() +
  facet_wrap(~cyl) +
  theme_cleaner(facets = TRUE, facet_text_size = 12, draw_panel_border = TRUE)
```

---

trauma_case_mix *View the Current Patient Population Case Mix Compared to the Major Trauma Study Case Mix*

Description

This function compares the current patient population's case mix (based on probability of survival, Ps) to the MTOS case mix by binning patients into specific Ps ranges. It returns the fraction of patients in each range and compares it to the MTOS distribution. For more information on the methods used in these calculations, please see Flora (1978) and Boyd et al. (1987).

Usage

```
trauma_case_mix(df, Ps_col, outcome_col)
```

Arguments

df	A data frame containing patient data.
Ps_col	The name of the column containing the probability of survival (Ps) values.
outcome_col	The name of the column containing the binary outcome data (valid values are 1 or TRUE for alive, 0 or FALSE for dead).

Details

The function checks whether the outcome_col contains exactly two unique values representing a binary outcome. It also ensures that Ps_col contains numeric values within the range 0 to 100. If any values exceed 1, they are converted to decimal format. The patients are then grouped into predefined Ps ranges, and the function compares the fraction of patients in each range with the MTOS case mix distribution.

Value

A data frame containing the Ps ranges, the fraction of patients in each range in the current population, and the MTOS distribution for each range.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Generate example data with high negative skewness
set.seed(123)

# Parameters
n_patients <- 10000  # Total number of patients

# Generate survival probabilities (Ps) using a logistic distribution
set.seed(123)  # For reproducibility
Ps <- plogis(rnorm(n_patients, mean = 2, sd = 1.5))  # Skewed towards higher values

# Simulate survival outcomes based on Ps
survival_outcomes <- rbinom(n_patients, size = 1, prob = Ps)

# Create data frame
data <- data.frame(Ps = Ps, survival = survival_outcomes) |>
dplyr::mutate(death = dplyr::if_else(survival == 1, 0, 1))

# Compare the current case mix with the MTOS case mix
trauma_case_mix(data, Ps_col = Ps, outcome_col = death)
```

---

trauma_performance *Calculate Trauma Hospital Performance Based on Robust and Validated Measures*

Description

This function calculates trauma hospital performance based on the M, W, and Z scores, which are derived from survival probability and mortality data, using established methods. It computes the W-score, M-score, and Z-score based on the provided dataset and calculates performance metrics for trauma programs. For more information on the methods used in this function, please see Champion et al. (1990) on the W score, and Flora (1978) and Boyd et al. (1987) on the M and Z scores.

Usage

```
trauma_performance(
  df,
  Ps_col,
  outcome_col,
  outcome = 1,
  z_method = c("survival", "mortality"),
  diagnostics = FALSE
)
```

Arguments

df	A data frame containing patient data.
Ps_col	The name of the column containing the probability of survival (Ps). The values should be numeric and between 0 and 1. Values greater than 1 will be automatically converted to decimal format by dividing by 100.
outcome_col	The name of the column containing the binary outcome data. The column should contain binary values indicating the patient outcome. Valid values include 1 (dead) and 0 (alive), or TRUE (dead) and FALSE (alive), or other similar binary representations (e.g., "Yes" for dead and "No" for alive). The function will check for two unique values in this column and expects them to represent the outcome in a binary form.
outcome	The value representing mortality (default is 1). Can also be set to 0 or TRUE/FALSE, depending on how the outcome is encoded in `outcome_col`.
z_method	A character vector indicating which method to use for calculating the Z-score. Must be one of "survival" or "mortality". The default is "survival".
diagnostics	A logical flag (default is FALSE). If TRUE, diagnostic information about the W, M, and Z scores will be printed to the console.

Value

A tibble containing the following calculations:

- `N_Patients`: The total number of patients included in the analysis.
- `N_Survivors`: The total number of patients who survived, based on the provided outcome data.
- `N_Deaths`: The total number of patients who died, based on the provided outcome data.
- `Predicted_Survivors`: The total predicted number of survivors based on the survival probability (`Ps`) for all patients.
- `Predicted_Deaths`: The total predicted number of deaths, calculated as `1 - Ps` for all patients.
- `Patient_Estimate`: The estimated number of patients who survived, calculated based on the W-score. This value reflects the difference between the actual and predicted number of survivors.
- `W_Score`: The W-score, representing the difference between the observed and expected number of survivors per 100 patients. A positive W-score indicates that more patients survived than expected, while a negative score indicates that fewer patients survived than expected.
- `M_Score`: The M-score, which compares the observed patient case mix to the Major Trauma Outcomes Study (MTOS) case mix. A higher score indicates that the patient mix is more similar to MTOS, while a lower score indicates a dissimilar mix. Based on the MTOS literature, an M_Score >= 0.88 indicates that the Z_Score comes from distribution similar enough to the MTOS Ps distribution.
- `Z_Score`: The Z-score, which quantifies the difference between the actual and predicted mortality (if `z_method = "mortality"`) or survival (if `z_method = "survival"`). A Z-score > 1.96 is considered to point to the statistical significance of the W-Score at alpha = 0.05 level for survival. The positive Z_Score indicates that more patients survived than predicted, while a negative Z-score indicates fewer survivors than predicted.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Generate example data with high negative skewness
set.seed(123)

# Parameters
n_patients <- 10000  # Total number of patients

# Generate survival probabilities (Ps) using a logistic distribution
set.seed(123)  # For reproducibility
Ps <- plogis(rnorm(n_patients, mean = 2, sd = 1.5))  # Skewed towards higher values

# Simulate survival outcomes based on Ps
survival_outcomes <- rbinom(n_patients, size = 1, prob = Ps)

# Create data frame
data <- data.frame(Ps = Ps, survival = survival_outcomes) |>
dplyr::mutate(death = dplyr::if_else(survival == 1, 0, 1))

# Calculate trauma performance (W, M, Z scores)
trauma_performance(data, Ps_col = Ps, outcome_col = death)
```

---

weekend *Classify Dates as Weekday or Weekend*

Description

This function classifies each date in a vector of dates as either "Weekday" or "Weekend". The function returns "Weekday" for Monday to Friday and "Weekend" for Saturday and Sunday.

Usage

```
weekend(input_date)
```

Arguments

`input_date`	A vector of `Date` or `POSIXct` objects to classify.

Details

The function checks if the `input_date` is a valid `Date` or `POSIXct` object. It returns "Weekday" for dates that fall on Monday through Friday and "Weekend" for dates that fall on Saturday or Sunday. If the input is not of the correct class, the function will throw an error.

Value

A character vector with the classification for each date: either "Weekday" or "Weekend".

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Example 1: Date of a weekend
weekend(as.Date("2025-01-18"))

# Example 2: Date of a weekday
weekend(as.Date("2025-01-15"))

# Example 3: Date of an invalid object
try(
weekend("2025-01-18") # This will throw an error
)
```

`%not_in%` *Check if Elements Are Not in a Vector*

Description

This function returns a logical vector indicating whether each element of `x` is not in `y`.

Usage

```
x %not_in% y
```

Arguments

`x`	A vector of values to be checked.
`y`	A vector of values to check against.

Value

A logical vector of the same length as `x`, where `TRUE` indicates the corresponding element in `x` is not found in `y`, and `FALSE` indicates it is found in `y`.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Example vectors
x <- c("apple", "banana", "cherry")
y <- c("banana", "grape")

# Check which elements in `x` are not in `y`
x %not_in% y

# Example with numeric values
a <- c(1, 2, 3, 4, 5)
b <- c(2, 4, 6)

a %not_in% b
```

Index

%not_in%, 23

impute, 2

nonlinear_bins, 3
normalize, 5

pretty_number, 6
pretty_percent, 7
probability_of_survival, 8

rm_bin_summary, 12
rmm, 9

season, 14
small_count_label, 15
stat_sig, 16

theme_cleaner, 17
trauma_case_mix, 19
trauma_performance, 20

weekend, 22