Package 'traumar'

April 16, 2026

Type Package

Title Calculate Metrics for Trauma System Performance

Version 1.2.5

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>. For more information about methods to calculate over- and under-triage in trauma hospital populations and samples, please see the following publications - Peng & Xiang (2016) <doi:10.1016/j.ajem.2016.08.061>, Beam et al. (2022) <doi:10.23937/2474-3674/1510136>, Roden-Foreman et al. (2017) <doi:10.1097/JTN.0000000000000283>.

License MIT + file LICENSE

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

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

Depends R (>= 4.1)

Imports cli, dplyr (>= 1.1.4), ggplot2 (>= 3.5.2), glue, hms (>= 1.1.3), infer (>= 1.0.8), lifecycle, lubridate (>= 1.9.4), nemsqar (>= 1.1.0), nortest (>= 1.0-4), patchwork (>= 1.3.1), purrr (>= 1.0.4), rlang, stats, stringr (>= 1.5.1), tibble (>= 3.3.0), tidyr (>= 1.3.1), tidyselect (>= 1.2.1), utils

Suggests broom, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

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 2026-04-16 21:50:02 UTC

Contents

`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")
```

---

is_it_normal *Exploratory Data Analysis, Normality Testing, and Visualization*

Description

[Experimental]

`is_it_normal()` calculates descriptive statistics and conducts univariate normality testing on one or more numeric variables in a dataset using a selected statistical test. Optional plots are included for one variable at a time, only. Results are returned as a named list containing summaries and, optionally, normality tests and/or diagnostic plots.

Usage

```
is_it_normal(
  df,
  ...,
  group_vars = NULL,
  seed = 10232015,
  normality_test = NULL,
  include_plots = FALSE,
  plot_theme = traumar::theme_cleaner
)
```

Arguments

`df` A `data.frame` or `tibble` containing the variables to assess.

`...` One or more unquoted column names from `df` to be analyzed.

group_vars	Optional. A character vector of column names in df to group results by (e.g., c("year", "hospital_level")). If NULL, no grouping is applied. Grouped summaries and normality tests are computed within each unique combination of values across these variables.
seed	A numeric value passed to set.seed() to ensure reproducibility. Default is 10232015.
normality_test	A character string specifying the statistical test to use. Must be one of: "shapiro-wilk" or "shapiro" o "kolmogorov-smirnov" or "ks", "anderson-darling" or "ad", "lilliefors" or "lilli", "cramer-von-mises" or "cvm", "pearson" or "p", or "shapiro-francia" or "sf". If NULL, no normality test is performed, which is the default.
include_plots	Logical. If TRUE, plots are generated for a single variable. Plotting is disabled if multiple variables are passed.
plot_theme	A ggplot2::theme function to apply to all plots. Default is traumar::theme_cleaner.

Details

- If the data do not meet the test requirements for a chosen test of normality, is_it_normal() will not run the tests.
- Normality tests may yield differing results. Each test has distinct assumptions and sensitivity. Users should verify assumptions and consult test-specific guidance to ensure appropriate use.
- The function will abort with helpful CLI messages if input types or structures are incorrect.
- If plotting is enabled, and nrow(df) > 10000, a warning is issued as plotting may become computationally expensive.

Value

A named list with the following elements:

descriptive_statistics A tibble of summary statistics for each variable.

normality_test A tibble of test statistics and p-values (if normality_test == TRUE).

plots A patchwork object containing four plots (if include_plots = TRUE and one variable supplied).

Note

Supported normality tests are below. Please check the specifications of these tests in the corresponding documentation.

- Shapiro-Wilk (stats::shapiro.test())
- Kolmogorov-Smirnov (stats::ks.test())
- Anderson-Darling (nortest::ad.test())
- Lilliefors (nortest::lillie.test())
- Cramer-von Mises (nortest::cvm.test())
- Pearson (norest::pearson.test())
- Shapiro-Francia (nortest::sf.test())

Please note that if grouped plotting is enabled, each group will generate its own set of plots. This may flood your IDE or console. Plan your use of this functionality with care to avoid lags or unwanted outputs.

Author(s)

Nicolas Foss, Ed.D., MS

`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,
  group_vars = NULL,
  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 name of the column containing the survival probabilities (Ps). Should be numeric on a scale from 0 to 1.
`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). `TRUE/FALSE` are accepted as well. Ensure the column contains only these possible values.
`group_vars`	Optional grouping variables for bin statistics calculations. These should be specified as quoted column names.
`divisor1`	A positive numeric value controlling the coarseness of bins for Ps values below `threshold_1`. It scales the number of steps from the start of the dataset up to the `threshold_1` cut point. Larger values produce fewer, broader bins; smaller values produce more, narrower bins. Defaults to `5`.

`divisor2` A positive numeric value controlling the coarseness of bins for Ps values between `threshold_1` and `threshold_2`. Larger values yield wider bins, and smaller values yield narrower bins in this range. Defaults to `5`.

`threshold_1` A numeric value that defines the lower bound of the high-survival probability range in `Ps_col`. The function identifies the first index where `Ps_col` exceeds this value and begins applying smaller bin widths from that point onward. Defaults to `0.9`, meaning binning changes once Ps > 0.90.

`threshold_2` A numeric value that defines the upper bound of the high-survival probability range in `Ps_col`. The function identifies the first index where `Ps_col` exceeds this value. Between `threshold_1` and `threshold_2`, finer binning is applied; above `threshold_2`, binning may again change. Defaults to `0.99`, meaning the special binning range is between Ps values of 0.90 and 0.99.

Details

Like other statistical computing functions, `nonlinear_bins()` is happiest without missing data. It is best to pass complete probability of survival and outcome data to the function for optimal performance. With smaller datasets, this is especially helpful. However, `nonlinear_bins()` will throw a warning about missing values, if any exist in `Ps_col` and/or `outcome_col`.

`nonlinear_bins()` assumes `Ps_col` contains probabilities derived from real-world inputs for the Trauma Injury Severity Score (TRISS) model. Synthetic or low-variability data (especially with small sample sizes) may not reflect the distribution of TRISS-derived survival probabilities. This can result in unstable estimates or function failure due to insufficient dispersion. With small sample sizes, it may be important to use smaller values with the divisor arguments and adjust the thresholds (based on the distribution of the `Ps_col` values) to create bins that better accommodate the data.

By default, `nonlinear_bins()` derives bin cut points from the full dataset's distribution. This ensures comparability across groups when `group_vars` is used. To tailor binning to a specific group (e.g., a single hospital), filter the dataset to that subgroup before calling `nonlinear_bins()`. The function will then compute bins and related statistics using only that subset's `Ps_col` distribution. When `group_vars` is used, and ff a group lacks observations within one or more bins, `rm_bin_summary()` will compute statistics only for the bins that contain data. Bins with no observations are excluded from the summary for that group.

The `threshold_1` and `threshold_2` arguments set probability cut points that define the start and end of a high-survival range where bin widths are adjusted for finer resolution. The `divisor1` and `divisor2` arguments are scaling factors that determine how many bins are created before and within this high-survival range, respectively. Lower divisors yield narrower bins, capturing more detail, while higher divisors yield broader bins, smoothing the distribution.

Value

A list with two elements:

- `intervals`: A vector defining bin boundaries for probability of survival.
- `bin_stats`: A `tibble` containing:
  - `bin_number`: Bin index.
  - `bin_start`, `bin_end`: Bin range.
  - `mean`, `sd`: Mean and standard deviation of `Ps_col` within the bin.

- `Pred_Survivors_b`, `Pred_Deaths_b`: Predicted counts of survivors and decedents, respectively.
- `AntiS_b`, `AntiM_b`: Anticipated proportion survived, and deceased, respectively.
- `alive`, `dead`: Count of observed survivors and non-survivors.
- `count`: Total records in the bin.
- `percent`: Percentage of total records within each bin.

Note

This function will produce the most reliable and interpretable results when using a dataset that has one row per patient, with each column being a feature.

The `mean` and `AntiS_b` are approximately equivalent in this context. They are kept in the output for clarity.

Author(s)

Nicolas Foss, Ed.D, MS, original implementation in MATLAB by Nicholas J. Napoli, Ph.D., MS

References

Kassar, O.M., Eklund, E.A., Barnhardt, W.F., Napoli, N.J., Barnes, L.E., Young, J.S. (2016). Trauma survival margin analysis: A dissection of trauma center performance through initial lactate. The American Surgeon, 82(7), 649-653. doi:10.1177/000313481608200733

Napoli, N. J., Barnhardt, W., Kotoriy, M. E., Young, J. S., & Barnes, L. E. (2017). Relative mortality analysis: A new tool to evaluate clinical performance in trauma centers. IISE Transactions on Healthcare Systems Engineering, 7(3), 181–191. doi:10.1080/24725579.2017.1325948

Schroeder, P. H., Napoli, N. J., Barnhardt, W. F., Barnes, L. E., & Young, J. S. (2018). Relative mortality analysis of the "golden hour": A comprehensive acuity stratification approach to address disagreement in current literature. Prehospital Emergency Care, 23(2), 254–262. doi:10.1080/10903127.2018.1489021

See Also

`probability_of_survival()`, `rmm()`, and `rm_bin_summary()`

Examples

```
# Generate example data
set.seed(123)

# Parameters
# Total number of patients
n_patients <- 5000

# Arbitrary group labels
groups <- sample(x = LETTERS[1:2], size = n_patients, replace = TRUE)

# Trauma types
trauma_type_values <- sample(
```

```
  x = c("Blunt", "Penetrating"),
  size = n_patients,
  replace = TRUE
)

# RTS values
rts_values <- sample(
  x = seq(from = 0, to = 7.8408, by = 0.005),
  size = n_patients,
  replace = TRUE
)

# patient ages
ages <- sample(
  x = seq(from = 0, to = 100, by = 1),
  size = n_patients,
  replace = TRUE
)

# ISS scores
iss_scores <- sample(
  x = seq(from = 0, to = 75, by = 1),
  size = n_patients,
  replace = TRUE
)

# Generate survival probabilities (Ps)
Ps <- traumar::probability_of_survival(
  trauma_type = trauma_type_values,
  age = ages,
  rts = rts_values,
  iss = iss_scores
)

# 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, groups = groups) |>
  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 = 4,
  divisor2 = 4,
  threshold_1 = 0.9,
  threshold_2 = 0.99
)

# View results
```

```
results$intervals
results$bin_stats

# Example with grouping by a categorical variable

# Run the function using a single grouping variable
results_grouped <- nonlinear_bins(
  data,
  Ps_col = Ps,
  outcome_col = survival,
  group_vars = "groups"
)

# View grouped results
results_grouped$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,
  digits = 2,
  n_decimal = deprecated(),
  prefix = NULL,
  truncate = FALSE
)
```

Arguments

`x`	A numeric value or vector to be converted into a readable format.
`digits`	Number of decimal places to display. Defaults to 2.
`n_decimal`	**[Deprecated]** Use `digits` instead.
`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. When `TRUE`, the function uses `base::signif()` to truncate the numbers to the specified number of significant digits, making the output more concise. When `FALSE`, the function uses `base::round()` to round the numbers to the specified number of decimal places, preserving the original scale of the number. 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, digits = 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).

Details

The methodology used in the calculation of survival probabilities aligns with the coefficients published in Norouzi et al. (2013) and Merchant et al. (2023). Consistent with Boyd et al. (1987), `probability_of_survival()` does not treat patients under 15 years of age differently and accounts for penetrating injuries similarly to other age groups. Norouzi et al. (2013) and Merchant et al. (2023) use the updated TRISS coefficients to calculate survival probabilities for penetrating traumas with the same coefficients as for blunt traumas. If this approach is preferred, please take note and adjust accordingly.

Value

Numeric vector of predicted probabilities of survival on a scale from 0 to 1.

Author(s)

Nicolas Foss, Ed.D., MS

References

Boyd CR, Tolson MA, Copes WS. (1987). Evaluating trauma care: the TRISS method. Trauma Score and the Injury Severity Score. J Trauma. 1987 Apr;27(4):370-8. PMID: 3106646.

Merchant AAH, Shaukat N, Ashraf N, Hassan S, Jarrar Z, Abbasi A, et al. (2023). Which curve is better? A comparative analysis of trauma scoring systems in a South Asian country. Trauma Surgery & Acute Care Open. 2023;8:e001171. doi:10.1136/tsaco-2023-001171

Norouzi V, Feizi I, Vatankhah S, Pourshaikhian M. (2013). Calculation of the probability of survival for trauma patients based on trauma score and the injury severity score model in fatemi hospital in ardabil. Arch Trauma Res. 2013 Spring;2(1):30-5. doi:10.5812/atr.9411. Epub 2013 Jun 1. PMID: 24396787; PMCID: PMC3876517.

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)
)

# Run the function on example data
result <- trauma_data |>
  dplyr::mutate(Ps = probability_of_survival(
    trauma_type = Trauma_Type,
    age = Patient_Age_Years,
    rts = RTS,
    iss = ISS
  ))

# Print the result
result
```

---

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.

Usage

```
rmm(
  data,
  Ps_col,
  outcome_col,
  group_vars = NULL,
  n_samples = 100,
  Divisor1 = 5,
  Divisor2 = 5,
  Threshold_1 = 0.9,
  Threshold_2 = 0.99,
  bootstrap_ci = TRUE,
  pivot = FALSE,
  seed = NULL
)
```

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 on a scale from 0 to 1.
`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). `TRUE/FALSE` are accepted as well. Ensure the column contains only these possible values.

group_vars — Optional character vector specifying grouping variables for stratified analysis. If NULL, the calculation is performed on the entire dataset.

n_samples — A numeric value indicating the number of bootstrap samples to take from the data source.

Divisor1 — A positive numeric value controlling the coarseness of bins for Ps values below Threshold_1. It scales the number of steps from the start of the dataset up to the Threshold_1 cut point. Larger values produce fewer, broader bins; smaller values produce more, narrower bins. Defaults to 5.

Divisor2 — A positive numeric value controlling the coarseness of bins for Ps values between Threshold_1 and Threshold_2. Larger values yield wider bins, and smaller values yield narrower bins in this range. Defaults to 5.

Threshold_1 — A numeric value that defines the lower bound of the high-survival probability range in Ps_col. The function identifies the first index where Ps_col exceeds this value and begins applying smaller bin widths from that point onward. Defaults to 0.9, meaning binning changes once Ps > 0.90.

Threshold_2 — A numeric value that defines the upper bound of the high-survival probability range in Ps_col. The function identifies the first index where Ps_col exceeds this value. Between Threshold_1 and Threshold_2, finer binning is applied; above Threshold_2, binning may again change. Defaults to 0.99, meaning the special binning range is between Ps values of 0.90 and 0.99.

bootstrap_ci — A logical indicating whether to return the relative mortality metric estimate and 95% confidence intervals using bootstrap sampling. Default is TRUE.

pivot — A logical indicating whether to return the results in a long format (pivot = TRUE) or wide format (pivot = FALSE, default). Use with caution in tandem with group_vars if the grouping variable is of a different class than rmm()'s outputs, such as factor or character grouping variables.

seed — Optional numeric value to set a random seed for reproducibility. If NULL (default), no seed is set.

Details

Like other statistical computing functions, rmm() is happiest without missing data. It is best to pass complete probability of survival and mortality outcome data to the function for optimal performance. With smaller datasets, this is especially helpful. However, rmm() will throw a warning about missing values, if any exist in Ps_col and/or outcome_col.

rmm() assumes Ps_col contains probabilities derived from real-world inputs for the Trauma Injury Severity Score (TRISS) model. Synthetic or low-variability data (especially with small sample sizes) may not reflect the distribution of TRISS-derived survival probabilities. This can result in unstable estimates or function failure due to insufficient dispersion. With small sample sizes, it may be important to use smaller values with the divisor arguments and adjust the thresholds (based on the distribution of the Ps_col values) to create bins that better accommodate the data.

Due to the use of bootstrap sampling within the function, users should consider setting the random number seed within rmm() for reproducibility.

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. (optional, if `bootstrap_ci = TRUE`)
- `bootstrap_RMM`: The average RMM value calculated for the bootstrap sample. (optional, if `bootstrap_ci = TRUE`)
- `bootstrap_RMM_UL`: The upper bound of the 95% confidence interval for the bootstrap RMM. (optional, if `bootstrap_ci = TRUE`)
- `bootstrap_CI`: The width of the 95% confidence interval for the bootstrap RMM. (optional, if `bootstrap_ci = TRUE`)
- 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.

Note

This function will produce the most reliable and interpretable results when using a dataset that has one row per patient, with each column being a feature.

By default, `rmm()` derives bin cut points from the full dataset's distribution. This ensures comparability across groups when `group_vars` is used. To tailor results to a specific group (e.g., a single hospital), filter the dataset to that subgroup before calling `rmm()`. The function will then compute bins and related statistics using only that subset's `Ps_col` distribution. When `group_vars` is used, and ff a group lacks observations within one or more bins, `rm_bin_summary()` will compute statistics only for the bins that contain data. Bins with no observations are excluded from the summary for that group.

Author(s)

Nicolas Foss, Ed.D, MS, original implementation in MATLAB by Nicholas J. Napoli, Ph.D., MS

References

Kassar, O.M., Eklund, E.A., Barnhardt, W.F., Napoli, N.J., Barnes, L.E., Young, J.S. (2016). Trauma survival margin analysis: A dissection of trauma center performance through initial lactate. The American Surgeon, 82(7), 649-653. doi:10.1177/000313481608200733

Napoli, N. J., Barnhardt, W., Kotoriy, M. E., Young, J. S., & Barnes, L. E. (2017). Relative mortality analysis: A new tool to evaluate clinical performance in trauma centers. IISE Transactions on Healthcare Systems Engineering, 7(3), 181–191. doi:10.1080/24725579.2017.1325948

Schroeder, P. H., Napoli, N. J., Barnhardt, W. F., Barnes, L. E., & Young, J. S. (2018). Relative mortality analysis of the "golden hour": A comprehensive acuity stratification approach to address disagreement in current literature. Prehospital Emergency Care, 23(2), 254–262. doi:10.1080/10903127.2018.1489021

See Also

probability_of_survival(), rm_bin_summary(), and nonlinear_bins()

Examples

```
# Generate example data
set.seed(123)

# Parameters
# Total number of patients
n_patients <- 5000

# Arbitrary group labels
groups <- sample(x = LETTERS[1:2], size = n_patients, replace = TRUE)

# Trauma types
trauma_type_values <- sample(
  x = c("Blunt", "Penetrating"),
  size = n_patients,
  replace = TRUE
)

# RTS values
rts_values <- sample(
  x = seq(from = 0, to = 7.8408, by = 0.005),
  size = n_patients,
  replace = TRUE
)

# patient ages
ages <- sample(
  x = seq(from = 0, to = 100, by = 1),
  size = n_patients,
  replace = TRUE
)

# ISS scores
iss_scores <- sample(
  x = seq(from = 0, to = 75, by = 1),
  size = n_patients,
  replace = TRUE
)
```

```
# Generate survival probabilities (Ps)
Ps <- traumar::probability_of_survival(
  trauma_type = trauma_type_values,
  age = ages,
  rts = rts_values,
  iss = iss_scores
)

# 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, groups = groups) |>
  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 = 4,
    Divisor2 = 4,
    n_samples = 10
    )

# pivot!
rmm(data = data, Ps_col = Ps,
    outcome_col = survival,
    Divisor1 = 4,
    Divisor2 = 4,
    n_samples = 10,
    pivot = TRUE
    )

# Create example grouping variable (e.g., hospital)
hospital <- sample(c("Hospital A", "Hospital B"), n_patients, replace = TRUE)

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

# Example usage of the `rmm` function with grouping by hospital
rmm(
  data = data,
  Ps_col = Ps,
  outcome_col = survival,
  group_vars = "hospital",
  Divisor1 = 4,
  Divisor2 = 4,
  n_samples = 10
)
```

```
# Pivoted output for easier visualization
rmm(
  data = data,
  Ps_col = Ps,
  outcome_col = survival,
  group_vars = "hospital",
  Divisor1 = 4,
  Divisor2 = 4,
  n_samples = 10,
  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.

Usage

```
rm_bin_summary(
  data,
  Ps_col,
  outcome_col,
  group_vars = NULL,
  n_samples = 100,
  Divisor1 = 5,
```

```
  Divisor2 = 5,
  Threshold_1 = 0.9,
  Threshold_2 = 0.99,
  bootstrap_ci = TRUE,
  seed = NULL
)
```

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 on a scale from 0 to 1.
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). TRUE/FALSE are accepted as well. Ensure the column contains only these possible values.
group_vars	Optional character vector specifying grouping variables for stratified analysis. If NULL, the calculation is performed on the entire dataset.
n_samples	A numeric value indicating the number of bootstrap samples to take from the data source.
Divisor1	A positive numeric value controlling the coarseness of bins for Ps values below Threshold_1. It scales the number of steps from the start of the dataset up to the Threshold_1 cut point. Larger values produce fewer, broader bins; smaller values produce more, narrower bins. Defaults to 5.
Divisor2	A positive numeric value controlling the coarseness of bins for Ps values between Threshold_1 and Threshold_2. Larger values yield wider bins, and smaller values yield narrower bins in this range. Defaults to 5.
Threshold_1	A numeric value that defines the lower bound of the high-survival probability range in Ps_col. The function identifies the first index where Ps_col exceeds this value and begins applying smaller bin widths from that point onward. Defaults to 0.9, meaning binning changes once Ps > 0.90.
Threshold_2	A numeric value that defines the upper bound of the high-survival probability range in Ps_col. The function identifies the first index where Ps_col exceeds this value. Between Threshold_1 and Threshold_2, finer binning is applied; above Threshold_2, binning may again change. Defaults to 0.99, meaning the special binning range is between Ps values of 0.90 and 0.99.
bootstrap_ci	A logical indicating whether to return the relative mortality metric estimate and 95% confidence intervals using bootstrap sampling. Default is TRUE.
seed	Optional numeric value to set a random seed for reproducibility. If NULL (default), no seed is set.

Details

Like other statistical computing functions, rm_bin_summary() is happiest without missing data. It is best to pass complete probability of survival and mortality outcome data to the function for optimal performance. With smaller datasets, this is especially helpful. However, rm_bin_summary() will throw a warning about missing values, if any exist in Ps_col and/or outcome_col.

rm_bin_summary() assumes Ps_col contains probabilities derived from real-world inputs for the Trauma Injury Severity Score (TRISS) model. Synthetic or low-variability data (especially with small sample sizes) may not reflect the distribution of TRISS-derived survival probabilities. This can result in unstable estimates or function failure due to insufficient dispersion. With small sample sizes, it may be important to use smaller values with the divisor arguments and adjust the thresholds (based on the distribution of the Ps_col values) to create bins that better accommodate the data.

Due to the use of bootstrap sampling within the function, users should consider setting the random number seed within rm_bin_summary() using the seed argument for reproducibility.

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. (optional, if bootstrap_ci = TRUE)
- bootstrap_RMM: The average RMM value calculated for the bootstrap sample. (optional, if bootstrap_ci = TRUE)
- bootstrap_RMM_UL: The upper bound of the 95% confidence interval for the bootstrap RMM. (optional, if bootstrap_ci = TRUE)
- bootstrap_CI: The width of the 95% confidence interval for the bootstrap RMM. (optional, if bootstrap_ci = TRUE)

Note

This function will produce the most reliable and interpretable results when using a dataset that has one row per patient, with each column being a feature.

By default, `rm_bin_summary()` derives bin cut points from the full dataset's distribution. This ensures comparability across groups when `group_vars` is used. To tailor results to a specific group (e.g., a single hospital), filter the dataset to that subgroup before calling `rm_bin_summary()`. The function will then compute bins and related statistics using only that subset's `Ps_col` distribution. When `group_vars` is used, and ff a group lacks observations within one or more bins, `rm_bin_summary()` will compute statistics only for the bins that contain data. Bins with no observations are excluded from the summary for that group.

Author(s)

Nicolas Foss, Ed.D, MS, original implementation in MATLAB by Nicholas J. Napoli, Ph.D., MS

References

Kassar, O.M., Eklund, E.A., Barnhardt, W.F., Napoli, N.J., Barnes, L.E., Young, J.S. (2016). Trauma survival margin analysis: A dissection of trauma center performance through initial lactate. The American Surgeon, 82(7), 649-653. doi:10.1177/000313481608200733

Napoli, N. J., Barnhardt, W., Kotoriy, M. E., Young, J. S., & Barnes, L. E. (2017). Relative mortality analysis: A new tool to evaluate clinical performance in trauma centers. IISE Transactions on Healthcare Systems Engineering, 7(3), 181–191. doi:10.1080/24725579.2017.1325948

Schroeder, P. H., Napoli, N. J., Barnhardt, W. F., Barnes, L. E., & Young, J. S. (2018). Relative mortality analysis of the "golden hour": A comprehensive acuity stratification approach to address disagreement in current literature. Prehospital Emergency Care, 23(2), 254–262. doi:10.1080/10903127.2018.1489021

See Also

`probability_of_survival()`, `rmm()`, and `nonlinear_bins()`

Examples

```
# Generate example data
set.seed(123)

# Parameters
# Total number of patients
n_patients <- 5000

# Arbitrary group labels
groups <- sample(x = LETTERS[1:2], size = n_patients, replace = TRUE)

# Trauma types
trauma_type_values <- sample(
  x = c("Blunt", "Penetrating"),
  size = n_patients,
  replace = TRUE
```

```
)

# RTS values
rts_values <- sample(
  x = seq(from = 0, to = 7.8408, by = 0.005),
  size = n_patients,
  replace = TRUE
)

# patient ages
ages <- sample(
  x = seq(from = 0, to = 100, by = 1),
  size = n_patients,
  replace = TRUE
)

# ISS scores
iss_scores <- sample(
  x = seq(from = 0, to = 75, by = 1),
  size = n_patients,
  replace = TRUE
)

# Generate survival probabilities (Ps)
Ps <- traumar::probability_of_survival(
  trauma_type = trauma_type_values,
  age = ages,
  rts = rts_values,
  iss = iss_scores
)

# 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, groups = groups) |>
  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 = 10,
  Divisor1 = 4,
  Divisor2 = 4
)

# Create example grouping variable (e.g., hospital)
hospital <- sample(c("Hospital A", "Hospital B"), n_patients, replace = TRUE)

# Create data frame
data <- data.frame(
```

```
  Ps = Ps,
  survival = survival_outcomes,
  hospital = hospital
) |>
  dplyr::mutate(death = dplyr::if_else(survival == 1, 0, 1))

# Example usage of the `rm_bin_summary()` function with grouping
rm_bin_summary(
  data = data,
  Ps_col = Ps,
  outcome_col = survival,
  group_vars = "hospital", # Stratifies by hospital
  n_samples = 10,
  Divisor1 = 4,
  Divisor2 = 4
)
```

`season` *Get Season Based on a Date*

Description

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

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).

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`.

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"))
```

`seqic_indicator_1` *SEQIC Indicator 1 – Trauma Team Response Evaluation*

Description

[Experimental]

This function calculates System Evaluation and Quality Improvement Committee (SEQIC) Indicator 1 (subparts a through f). These indicators assess the timeliness and type of provider response (e.g., surgeon, mid-level, physician) to trauma alerts based on trauma team activation level, hospital trauma level, and time to provider presence. Confidence intervals can optionally be calculated for the proportion, using either the Wilson or Clopper-Pearson method.

Usage

```
seqic_indicator_1(
  data,
  trauma_team_activation_level,
  trauma_team_physician_service_type,
  level,
  included_levels = c("I", "II", "III", "IV"),
  unique_incident_id,
  response_time,
  trauma_team_activation_provider,
  groups = NULL,
  calculate_ci = NULL,
  ...
)
```

Arguments

`data`	A data frame containing trauma incident records.
`trauma_team_activation_level`	Column identifying trauma team activation level (e.g., Level 1, Level 2).
`trauma_team_physician_service_type`	Column indicating the type of medical provider (e.g., Surgery/Trauma, Emergency Medicine). For indicators 1a, 1b, and 1c, `seqic_indicator_1()` will

only look for records with the trauma team member service type documented as marked as 'Surgery/Trauma'. For Indicators 1d, 1e, and 1f, `seqic_indicator_1()` will look for the following service types:

- "Surgery/Trauma",
- "Emergency Medicine",
- "Family Practice",
- "Nurse Practitioner",
- "Physician Assistant",
- "Surgery Senior Resident",
- "Hospitalist",
- "Internal Medicine"

`level` Column indicating the trauma center designation level (e.g., I, II, III, IV).

`included_levels`
Character vector indicating what facility levels to include in the analysis. Defaults to `c("I", "II", "III", "IV")`.

`unique_incident_id`
Unique identifier for each record.

`response_time` Numeric variable representing the time (in minutes) to provider response.

`trauma_team_activation_provider`
Column identifying the responding provider for trauma activation.

`groups` Additional columns passed as a vector of strings to `dplyr::summarize()` via the `.by` argument for grouped summaries. Defaults to `NULL`.

`calculate_ci` If `NULL`, 95% confidence intervals will not be calculated for the performance estimates. Otherwise, options of "wilson" or "clopper-pearson" can be supplied to utilize the corresponding methods to calculate the confidence intervals for the proportions. Defaults to `NULL`.

`...` Arguments passed on to `nemsqar::nemsqa_binomial_confint`

`conf.level` Numeric value between 0 and 1 indicating the confidence level. Defaults to 0.95 (95% confidence interval).

`correct` Logical, indicating whether to apply continuity correction for Wilson intervals. Defaults to `TRUE`.

Details

This function filters and summarizes trauma records to calculate SEQIC Indicators 1a through 1f:

- 1a: Proportion of Level 1 activations at Level I/II centers with surgical response <= 15 minutes.
- 1b: Same as 1a, but includes Level III centers and uses <= 30 minutes.
- 1c: Proportion of Level 1 activations with missing surgical response time.
- 1d/e: Response within 5 and 20 minutes, respectively, for specific provider types and activation levels, includes level I-IV trauma centers.
- 1f: Proportion of missing response times among the group in 1d/e, includes level I-IV trauma centers.

Value

A tibble summarizing SEQIC Indicator 1 results across sub-measures (1a–1f). Includes numerators, denominators, and performance rate for each indicator. 95% confidence intervals are provided optionally.

Note

This function:

- Filters trauma records to those with a trauma team activation level of "Level 1" and/or "Level 2" based on the indicator.
- Restricts provider type to surgical, physician, and mid-level provider roles.
- Filters trauma center levels to I–IV based on the measure.
- Calculates the proportion of cases where the response time is within 5, 15, or 30 minutes, depending on the indicator.
- Computes proportions for trauma activation times, including missing times and within thresholds.

Users must ensure appropriate column names are passed and data is pre-processed to include the necessary fields without missing critical identifiers or timestamps.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Packages
library(dplyr)
library(traumar)

# Data
data <- tibble::tibble(
  incident_id = 1:6,
  activation_level = c("Level 1", "Level 1", "Level 2", "Level 1", "Level 2",
  "Level 1"),
  provider_type = c("Surgery/Trauma", "Emergency Medicine", "Physician
  Assistant", "Surgery/Trauma", "Surgery/Trauma", "Family Practice"),
  trauma_level = c("I", "II", "III", "I", "III", "IV"),
  response_minutes = c(12, 25, 6, NA, 18, 22),
  provider = c("Dr. A", "Dr. B", "PA C", "Dr. D", "Dr. E", "NP F")
)

# Run the function
traumar::seqic_indicator_1(
  data = data,
  trauma_team_activation_level = activation_level,
  trauma_team_physician_service_type = provider_type,
  level = trauma_level,
  unique_incident_id = incident_id,
```

```
  response_time = response_minutes,
  trauma_team_activation_provider = provider,
  calculate_ci = "wilson"
) |>
tidyr::pivot_longer(cols = -1,
                    names_to = "Indicator",
                    values_to = "Values"
                    )
```

`seqic_indicator_10` *SEQIC Indicator 10 – Trauma Team Activation Appropriateness*

Description

[Experimental]

Calculates three trauma system quality indicators related to trauma team activations where the patient was kept at the facility:

- 10a: Proportion of patients meeting triage criteria (based on Injury Severity Score or Need For Trauma Intervention) who received low-level or no activation (undertriage).
- 10b: Proportion of patients not meeting triage criteria who received highest-level trauma activation (overtriage).
- 10c: Proportion of major trauma patients receiving a full activation (undertriage via Peng & Xiang, 2019).

  (10a, 10b, 10c can be based on Injury Severity Score or Need For Trauma Intervention based on user choice)

Users may stratify results by one or more grouping variables and optionally compute confidence intervals.

Usage

```
seqic_indicator_10(
  data,
  level,
  included_levels = c("I", "II", "III", "IV"),
  unique_incident_id,
  transfer_out_indicator,
  trauma_team_activation_level,
  iss,
  nfti,
  groups = NULL,
  calculate_ci = NULL,
  ...
)
```

Arguments

`data`	A data frame containing trauma incident records.
`level`	Column indicating the trauma center designation level (e.g., I, II, III, IV).
`included_levels`	Character vector indicating what facility levels to include in the analysis. Defaults to `c("I", "II", "III", "IV")`.
`unique_incident_id`	Unique identifier for each record.
`transfer_out_indicator`	Column name indicating whether the patient was transferred out of the initial trauma center to definitive care. Logical, character, or factor type. Values representing "No" (e.g., FALSE, "No") indicate no transfer out.
`trauma_team_activation_level`	Column indicating the trauma team activation level (e.g., `"Level 1"`, `"Level 2"`, `"Level 3"`, `"Consultation"`). Must be character or factor.
`iss`	Optional numeric column representing the Injury Severity Score.
`nfti`	Optional column indicating Need For Trauma Intervention classification of positive or negative. Should be character, factor, or logical.
`groups`	Optional character vector of column names used for grouping results.
`calculate_ci`	Optional; if not `NULL`, must be `"wilson"` or `"clopper-pearson"` to compute confidence intervals.
`...`	Arguments passed on to `nemsqar::nemsqa_binomial_confint` `conf.level` Numeric value between 0 and 1 indicating the confidence level. Defaults to 0.95 (95% confidence interval). `correct` Logical, indicating whether to apply continuity correction for Wilson intervals. Defaults to `TRUE`.

Details

This function:

- Restricts analysis to Level I–IV trauma centers.
- Removes duplicate incidents using `unique_incident_id`.
- Classifies each record as meeting or not meeting triage criteria based on ISS or NFTI logic.
- Optionally computes 95% confidence intervals for each indicator.

Users must ensure appropriate column names are passed and data is pre-processed to include the necessary fields without missing critical identifiers or timestamps.

Value

A list of two tibbles with counts and proportions for SEQIC Indicators 10a, 10b, and 10c, along with model diagnostics for the Cribari or NFTI ouputs. The proportions in 10a, 10b, and 10c will optionally include 95% confidence intervals.

Author(s)

Nicolas Foss, Ed.D., MS

References

Beam G, Gorman K, Nannapaneni S, Zipf J, Simunich T, et al. (2022) Need for Trauma Intervention and Improving Under-Triaging in Geriatric Trauma Patients: undertriaged or Misclassified. Int J Crit Care Emerg Med 8:136. doi.org/10.23937/2474-3674/1510136

Peng J, Xiang H. Trauma undertriage and overtriage rates: are we using the wrong formulas? Am J Emerg Med. 2016 Nov;34(11):2191-2192. doi: 10.1016/j.ajem.2016.08.061. Epub 2016 Aug 31. PMID: 27615156; PMCID: PMC6469681.

Roden-Foreman JW, Rapier NR, Yelverton L, Foreman ML. Asking a Better Question: Development and Evaluation of the Need For Trauma Intervention (NFTI) Metric as a Novel Indicator of Major Trauma. J Trauma Nurs. 2017 May/Jun;24(3):150-157. doi: 10.1097/JTN.0000000000000283. PMID: 28486318.

Examples

```
# Packages
library(dplyr)
library(traumar)

# Simulated data for SEQIC Indicator 10
test_data <- tibble::tibble(
  id = as.character(1:12),
  trauma_level = c("I", "II", "III", "IV", "II", "I", "IV", "III", "II", "I",
  "III", "IV"),
  activation = c("Level 1", "Level 2", "None", "Consultation", "Level 1",
  "Level 1", "None", "Level 3", "Level 1", "Consultation", "None", "Level
  2"),
  acute_transfer = rep("No", 12),
  iss = c(25, 10, 16, 8, 30, 45, 12, 9, 28, 6, 17, 14),
  nfti = c(TRUE, FALSE, TRUE, FALSE, TRUE, TRUE, FALSE, FALSE, TRUE, FALSE,
  TRUE, TRUE),
  region = rep(c("East", "West"), each = 6)
)

# Run the function, this will succeed
traumar::seqic_indicator_10(
  data = test_data,
  level = trauma_level,
  included_levels = c("I", "II", "III", "IV"),
  unique_incident_id = id,
  transfer_out_indicator = acute_transfer,
  trauma_team_activation_level = activation,
  iss = iss,
  nfti = NULL,
  groups = "region",
  calculate_ci = "wilson"
)
```

```
# Run the function, this will fail
# Only one of `iss` or `nfti` arguments can be passed, not both
try(
  traumar::seqic_indicator_10(
  data = test_data,
  level = trauma_level,
  included_levels = c("I", "II", "III", "IV"),
  unique_incident_id = id,
  transfer_out_indicator = acute_transfer,
  trauma_team_activation_level = activation,
  iss = iss,
  nfti = nfti,
  groups = "region",
  calculate_ci = "wilson"
))
```

`seqic_indicator_11` *SEQIC Indicator 11 – Overtriage for Minor Trauma Patients*

Description

[Experimental]

Calculates SEQIC Indicator 11, which estimates the proportion of minor trauma patients who were transferred into a trauma center and remained in the Emergency Department for less than 24 hours. This indicator is designed to identify potential overtriage events within the trauma system. Minor trauma patients are identified using the Injury Severity Score (ISS < 9). Patients must not have been transferred out and must have been received at a trauma center level included in `included_levels`.

Usage

```
seqic_indicator_11(
  data,
  level,
  included_levels = c("I", "II", "III", "IV"),
  transfer_out_indicator,
  receiving_indicator,
  unique_incident_id,
  iss,
  ed_LOS,
  groups = NULL,
  calculate_ci = NULL,
  ...
)
```

Arguments

`data`	A data frame containing trauma incident records.
`level`	Column indicating the trauma center designation level (e.g., I, II, III, IV).
`included_levels`	Character vector indicating what facility levels to include in the analysis. Defaults to `c("I", "II", "III", "IV")`.
`transfer_out_indicator`	Column name indicating whether the patient was transferred out of the initial trauma center to definitive care. Logical, character, or factor type. Values representing "No" (e.g., FALSE, "No") indicate no transfer out.
`receiving_indicator`	Column name indicating whether the patient was transferred into the trauma center. Logical, character, or factor type. Values representing "Yes" (e.g., TRUE, "Yes") indicate transfer in.
`unique_incident_id`	Unique identifier for each record.
`iss`	Optional numeric column representing the Injury Severity Score.
`ed_LOS`	Column for the calculated ED length of stay, measured in minutes.
`groups`	Additional columns passed as a vector of strings to `dplyr::summarize()` via the `.by` argument for grouped summaries. Defaults to `NULL`.
`calculate_ci`	If `NULL`, 95% confidence intervals will not be calculated for the performance estimates. Otherwise, options of "wilson" or "clopper-pearson" can be supplied to utilize the corresponding methods to calculate the confidence intervals for the proportions. Defaults to `NULL`.
`...`	Arguments passed on to `nemsqar::nemsqa_binomial_confint` `conf.level` Numeric value between 0 and 1 indicating the confidence level. Defaults to 0.95 (95% confidence interval). `correct` Logical, indicating whether to apply continuity correction for Wilson intervals. Defaults to `TRUE`.

Details

This function:

- Filters the dataset to include only patients treated at trauma centers designated Level I through IV.
- Excludes patients transferred out and retains only those received by the trauma center.
- Deduplicates incident-level records using `unique_incident_id`.
- Classifies patients as low-risk based on the Injury Severity Score (ISS < 9).
- Flags low-risk patients who were discharged from the ED in under 24 hours.
- Stratifies results by one or more user-defined grouping variables.
- Returns a summarized tibble with the number of eligible low-risk short-stay discharges (numerator), all received patients meeting inclusion criteria (denominator), and the resulting proportion.

- Optionally includes 95% confidence intervals if `calculate_ci` is specified.

Users must ensure appropriate column names are passed and data is pre-processed to include the necessary fields without missing critical identifiers or timestamps.

Value

A tibble summarizing the numerator, denominator, and proportion of overtriaged patients (Indicator 11), with optional 95% confidence intervals.

Author(s)

Nicolas Foss, Ed.D., MS

References

Roden-Foreman JW, Rapier NR, Yelverton L, Foreman ML. Asking a Better Question: Development and Evaluation of the Need For Trauma Intervention (NFTI) Metric as a Novel Indicator of Major Trauma. J Trauma Nurs. 2017 May/Jun;24(3):150-157. doi: 10.1097/JTN.0000000000000283. PMID: 28486318.

Examples

```
# Packages
library(dplyr)
library(traumar)

# Simulated data for SEQIC Indicator 11
test_data <- tibble::tibble(
  id = as.character(1:10),
  trauma_level = c("I", "II", "III", "IV", "II", "I", "IV", "III", "II",
  "I"),
  transferred_out = c(FALSE, FALSE, FALSE, TRUE, FALSE, FALSE, TRUE, FALSE,
  FALSE, FALSE),
  received = c(TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE),
  iss = c(4, 8, 10, 6, 5, 7, 6, 15, 3, 2),
  ed_LOS = c(6, 20, 30, 18, 8, 5, 22, 40, 2, 4),
  region = rep(c("East", "West"), each = 5)
)

# Run the function
traumar::seqic_indicator_11(
  data = test_data,
  level = trauma_level,
  included_levels = c("I", "II", "III", "IV"),
  transfer_out_indicator = transferred_out,
  receiving_indicator = received,
  unique_incident_id = id,
  iss = iss,
  ed_LOS = ed_LOS,
  groups = "region",
  calculate_ci = "clopper-pearson"
```

```
)
```

`seqic_indicator_12` *SEQIC Indicator 12 - Timeliness of Data Entry Post-Discharge*

Description

[Experimental]

Calculates the proportion of trauma cases where data were entered into the trauma registry within a defined number of days post-discharge. This measure supports trauma system quality improvement by identifying facilities meeting timely documentation expectations.

Usage

```
seqic_indicator_12(
  data,
  level,
  included_levels = c("I", "II", "III", "IV"),
  facility_id,
  exclude_facility_list = NULL,
  unique_incident_id,
  data_entry_time,
  data_entry_standard = 60,
  groups = NULL,
  calculate_ci = NULL,
  ...
)
```

Arguments

`data`	A data frame containing trauma incident records.
`level`	Column indicating the trauma center designation level (e.g., I, II, III, IV).
`included_levels`	Character vector indicating what facility levels to include in the analysis. Defaults to `c("I", "II", "III", "IV")`.
`facility_id`	Numeric, character, or factor. Column giving the unique facility identifiers in the trauma dataset.
`exclude_facility_list`	Optional. Numeric, character, or factor. List of facilities to exclude from analysis due to known data quality issues or other justifiable reasons. Defaults to `NULL`.
`unique_incident_id`	Unique identifier for each record.
`data_entry_time`	Numeric. Column representing the time in days between patient discharge and trauma registry data entry.

`data_entry_standard`
: Numeric. The maximum allowable number of days between discharge and data entry. Records entered within this threshold are considered timely. Default is `60`.

`groups`
: Additional columns passed as a vector of strings to `dplyr::summarize()` via the `.by` argument for grouped summaries. Defaults to `NULL`.

`calculate_ci`
: If `NULL`, 95% confidence intervals will not be calculated for the performance estimates. Otherwise, options of "wilson" or "clopper-pearson" can be supplied to utilize the corresponding methods to calculate the confidence intervals for the proportions. Defaults to `NULL`.

`...`
: Arguments passed on to `nemsqar::nemsqa_binomial_confint`

  `conf.level` Numeric value between 0 and 1 indicating the confidence level. Defaults to 0.95 (95% confidence interval).

  `correct` Logical, indicating whether to apply continuity correction for Wilson intervals. Defaults to `TRUE`.

Details

This function:

- Filters to include only patients treated at Level I–IV trauma centers.
- Excludes records from facilities specified by the user, if applicable.
- Deduplicates by `unique_incident_id` to ensure each incident is counted once.
- Flags records where data entry occurred within `data_entry_standard` days of discharge.
- Optionally calculates confidence intervals using methods from `nemsqa_binomial_confint()`.
- Returns a tibble with numerator, denominator, and proportion of timely entries, with optional confidence intervals and population/sample labels.

Users must ensure appropriate column names are passed and data is pre-processed to include the necessary fields without missing critical identifiers or timestamps.

Value

A tibble summarizing SEQIC Indicator 12 results. Includes numerator, denominator, and performance rate. 95% confidence intervals are included if requested.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Packages
library(dplyr)
library(traumar)

# Simulated data for SEQIC Indicator 12
test_data <- tibble::tibble(
```

```
  id = as.character(1:10),
  trauma_level = c("I", "II", "III", "IV", "II", "I", "IV", "III", "II",
  "I"),
  facility = c("A", "B", "C", "D", "A", "C", "B", "A", "C", "D"),
  data_entry_delay = c(30, 65, 10, 70, 45, 20, 80, 15, 55, 90)
)

# Run the function
traumar::seqic_indicator_12(
  data = test_data,
  level = trauma_level,
  included_levels = c("I", "II", "III", "IV"),
  facility_id = facility,
  unique_incident_id = id,
  exclude_facility_list = c("D"),
  data_entry_time = data_entry_delay,
  data_entry_standard = 60,
  calculate_ci = "wilson"
)
```

`seqic_indicator_13` *SEQIC Indicator 13 – Validation of Trauma Registry Records*

Description

[Experimental]

Calculates the proportion of trauma records that meet or exceed a threshold for data validity among facilities at the specified trauma center levels. Optionally computes confidence intervals.

Usage

```
seqic_indicator_13(
  data,
  level,
  included_levels = c("I", "II", "III", "IV"),
  unique_incident_id,
  validity_score,
  validity_threshold = 85,
  groups = NULL,
  calculate_ci = NULL,
  ...
)
```

Arguments

`data`	A data frame containing trauma incident records.
`level`	Column indicating the trauma center designation level (e.g., I, II, III, IV).

`included_levels`
: Character vector indicating what facility levels to include in the analysis. Defaults to `c("I", "II", "III", "IV")`.

`unique_incident_id`
: Unique identifier for each record.

`validity_score`
: Numeric. The proportion of each trauma registry record that is valid, expressed as a percentage (0–100). Typically calculated by the registry system.

`validity_threshold`
: Numeric. The minimum acceptable validity percentage threshold for records to be counted in the numerator. Defaults to `85`.

`groups`
: Additional columns passed as a vector of strings to `dplyr::summarize()` via the `.by` argument for grouped summaries. Defaults to `NULL`.

`calculate_ci`
: If `NULL`, 95% confidence intervals will not be calculated for the performance estimates. Otherwise, options of "wilson" or "clopper-pearson" can be supplied to utilize the corresponding methods to calculate the confidence intervals for the proportions. Defaults to `NULL`.

`...`
: Arguments passed on to `nemsqar::nemsqa_binomial_confint`

  `conf.level` Numeric value between 0 and 1 indicating the confidence level. Defaults to 0.95 (95% confidence interval).

  `correct` Logical, indicating whether to apply continuity correction for Wilson intervals. Defaults to `TRUE`.

Details

This function:

- Filters to include only patients treated at trauma centers with levels specified in `included_levels` (default: Levels I–IV).
- Deduplicates the dataset using `unique_incident_id` to ensure each incident is counted only once.
- Flags records with a `validity_score` greater than or equal to the specified `validity_threshold` threshold (default: 85).
- Calculates the proportion of valid records among all included records.
- Optionally calculates binomial confidence intervals using the method specified in `calculate_ci` via `nemsqa_binomial_confint()`.
- Adds a "Population/Sample" label unless grouping is applied via `groups`.

Users must ensure that appropriate column names are passed using tidy evaluation (bare column names) and that the input data has been cleaned and includes no missing or malformed identifiers, trauma level classifications, or validity scores.

Value

A tibble summarizing SEQIC Indicator 13 results. Includes numerator, denominator, and performance rate 95% confidence intervals are included if requested.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Packages
library(dplyr)
library(traumar)

# Simulated data for SEQIC Indicator 13
test_data <- tibble::tibble(
  id = as.character(1:12),
  trauma_level = c("I", "II", "III", "IV", "I", "II", "III", "IV", "I", "II",
  "III", "IV"),
  validity = c(90, 80, 88, 92, 86, 75, 89, 70, 95, 85, 83, 87)
)

# Run the function
traumar::seqic_indicator_13(
  data = test_data,
  level = trauma_level,
  included_levels = c("I", "II", "III", "IV"),
  unique_incident_id = id,
  validity_score = validity,
  validity_threshold = 85,
  calculate_ci = "wilson"
)
```

---

seqic_indicator_2 *SEQIC Indicator 2 – Missing Incident Time*

Description

[Experimental]

This function calculates System Evaluation and Quality Improvement Committee (SEQIC) Indicator 2. This indicator evaluates the proportion of trauma incidents with missing documented incident time across Level I–IV trauma centers.

Usage

```
seqic_indicator_2(
  data,
  unique_incident_id,
  level,
  included_levels = c("I", "II", "III", "IV"),
  incident_time,
  groups = NULL,
```

```
  calculate_ci = NULL,
  ...
)
```

Arguments

`data`	A data frame containing trauma incident records.
`unique_incident_id`	Unique identifier for each record.
`level`	Column indicating the trauma center designation level (e.g., I, II, III, IV).
`included_levels`	Character vector indicating what facility levels to include in the analysis. Defaults to `c("I", "II", "III", "IV")`.
`incident_time`	The time the patient's injury occurred.
`groups`	Additional columns passed as a vector of strings to `dplyr::summarize()` via the `.by` argument for grouped summaries. Defaults to `NULL`.
`calculate_ci`	If `NULL`, 95% confidence intervals will not be calculated for the performance estimates. Otherwise, options of "wilson" or "clopper-pearson" can be supplied to utilize the corresponding methods to calculate the confidence intervals for the proportions. Defaults to `NULL`.
`...`	Arguments passed on to `nemsqar::nemsqa_binomial_confint` `conf.level` Numeric value between 0 and 1 indicating the confidence level. Defaults to 0.95 (95% confidence interval). `correct` Logical, indicating whether to apply continuity correction for Wilson intervals. Defaults to `TRUE`.

Details

This function:

- Filters trauma records to those with a trauma center level of I–IV.
- Deduplicates by `unique_incident_id` to ensure one record per incident.
- Calculates the proportion of cases missing `incident_time`.

Value

A tibble summarizing SEQIC Indicator 2 results. Includes numerator, denominator, and performance rate for the indicator. 95% confidence intervals are provided optionally.

Note

Users must ensure appropriate column names are passed and data is pre-processed to include the necessary fields without missing critical identifiers or timestamps.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Packages
library(dplyr)
library(traumar)

# Data
data <- tibble::tibble(
  incident_id = as.character(101:106),
  trauma_level = c("I", "II", "III", "IV", "II", "I"),
  incident_time = as.POSIXct(c("2023-01-01 12:00", NA, "2023-01-02 14:15",
                               NA, "2023-01-03 09:30", "2023-01-04 16:45"))
)

# Run the function
traumar::seqic_indicator_2(
  data = data,
  unique_incident_id = incident_id,
  level = trauma_level,
  incident_time = incident_time,
  calculate_ci = "clopper-pearson"
)
```

---

seqic_indicator_3 *SEQIC Indicator 3 - Presence of Probability of Survival Calculations*

Description

[Experimental]

This function calculates Indicator 3, a measure of the proportion of trauma incidents where the probability of survival is recorded. It filters the data by trauma center level (I-IV), excluding burn cases, and computes the proportion of incidents with a valid probability of survival value.

Usage

```
seqic_indicator_3(
  data,
  level,
  included_levels = c("I", "II", "III", "IV"),
  trauma_type,
  unique_incident_id,
  probability_of_survival,
  groups = NULL,
  calculate_ci = NULL,
  ...
)
```

Arguments

`data`	A data frame containing trauma incident records.
`level`	Column indicating the trauma center designation level (e.g., I, II, III, IV).
`included_levels`	Character vector indicating what facility levels to include in the analysis. Defaults to `c("I", "II", "III", "IV")`.
`trauma_type`	A column name indicating the type of trauma. The function filters out "Burn" cases.
`unique_incident_id`	Unique identifier for each record.
`probability_of_survival`	A column name for the probability of survival for each incident.
`groups`	Additional columns passed as a vector of strings to `dplyr::summarize()` via the `.by` argument for grouped summaries. Defaults to `NULL`.
`calculate_ci`	If `NULL`, 95% confidence intervals will not be calculated for the performance estimates. Otherwise, options of "wilson" or "clopper-pearson" can be supplied to utilize the corresponding methods to calculate the confidence intervals for the proportions. Defaults to `NULL`.
`...`	Arguments passed on to `nemsqar::nemsqa_binomial_confint` `conf.level` Numeric value between 0 and 1 indicating the confidence level. Defaults to 0.95 (95% confidence interval). `correct` Logical, indicating whether to apply continuity correction for Wilson intervals. Defaults to `TRUE`.

Details

This function:

- Filters trauma records to those with a trauma center level of I–IV.
- Excludes records with a trauma type of "Burn".
- Deduplicates by `unique_incident_id` to ensure one record per incident.
- Calculates the proportion of records with a non-missing `probability_of_survival`.

Value

A tibble summarizing SEQIC Indicator 3 results. Includes numerator, denominator, and performance rate for the indicator. 95% confidence intervals are provided optionally.

Note

Users must ensure appropriate column names are passed and data is pre-processed to include the necessary fields without missing critical identifiers or timestamps.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Packages
library(dplyr)
library(traumar)

# Create a synthetic test dataset
test_data <- tibble::tibble(
  unique_id = as.character(1:10),
  trauma_level = c("I", "II", "III", "IV", "I", "II", "III", "IV", "I",
  "II"),
  trauma_category = c("Blunt", "Penetrating", "Burn", "Blunt", "Penetrating",
                      "Burn", "Blunt", "Penetrating", "Blunt", "Blunt"),
  survival_prob = c(0.95, 0.89, NA, 0.76, NA, 0.92, 0.88, NA, 0.97, 0.91)
)

# Run the indicator function
traumar::seqic_indicator_3(
  data = test_data,
  level = trauma_level,
  trauma_type = trauma_category,
  unique_incident_id = unique_id,
  probability_of_survival = survival_prob,
  groups = "trauma_level"
)
```

---

seqic_indicator_4 *SEQIC Indicator 4 - Autopsy and Long LOS Without Autopsy*

Description

[Experimental]

Computes SEQIC Indicator 4a and 4b for trauma center performance. Indicator 4a captures the proportion of deceased trauma patients at trauma level I–IV facilities who had an autopsy performed. Indicator 4b identifies deceased trauma patients with a prolonged length of stay (LOS > 3 days) but without an autopsy.

Usage

```
seqic_indicator_4(
  data,
  level,
  included_levels = c("I", "II", "III", "IV"),
  ed_disposition,
  ed_LOS,
  hospital_disposition,
  hospital_LOS,
  unique_incident_id,
```

```
  autopsy,
  groups = NULL,
  calculate_ci = NULL,
  ...
)
```

Arguments

`data`	A data frame containing trauma incident records.
`level`	Column indicating the trauma center designation level (e.g., I, II, III, IV).
`included_levels`	Character vector indicating what facility levels to include in the analysis. Defaults to `c("I", "II", "III", "IV")`.
`ed_disposition`	Column representing the emergency department disposition. For a record to be picked up in this function, the ED dispostion must be documented as "Deceased/Expired".
`ed_LOS`	Column for the calculated ED length of stay, measured in minutes.
`hospital_disposition`	Column representing the hospital disposition. For a record to be picked up in this function, the hospital dispostion must be documented as "Deceased/Expired".
`hospital_LOS`	Column for the calculated hospital length of stay, measured in minutes.
`unique_incident_id`	Unique identifier for each record.
`autopsy`	Unquoted column name indicating whether an autopsy was performed. Expected values: `"Yes"` or `NA`.
`groups`	Additional columns passed as a vector of strings to `dplyr::summarize()` via the `.by` argument for grouped summaries. Defaults to `NULL`.
`calculate_ci`	If `NULL`, 95% confidence intervals will not be calculated for the performance estimates. Otherwise, options of "wilson" or "clopper-pearson" can be supplied to utilize the corresponding methods to calculate the confidence intervals for the proportions. Defaults to `NULL`.
`...`	Arguments passed on to `nemsqar::nemsqa_binomial_confint` `conf.level` Numeric value between 0 and 1 indicating the confidence level. Defaults to 0.95 (95% confidence interval). `correct` Logical, indicating whether to apply continuity correction for Wilson intervals. Defaults to `TRUE`.

Details

This function:

- Filters trauma records to those with a trauma center level of I–IV.
- Identifies records where the patient died, based on ED or hospital disposition.
- Deduplicates by `unique_incident_id` to ensure one record per incident.
- For Indicator 4a, calculates the proportion of deceased patients who received an autopsy.
- For Indicator 4b, calculates the proportion of deceased patients with a hospital or ED length of stay greater than 72 hours (4320 minutes) and no autopsy performed.

Value

A tibble summarizing SEQIC Indicator 4a and 4b results. Includes numerator, denominator, and performance rate for the indicator. 95% confidence intervals are provided optionally.

Note

Users must ensure appropriate column names are passed and data is pre-processed to include the necessary fields without missing critical identifiers or timestamps.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Packages
library(dplyr)
library(traumar)

# Create a synthetic test dataset
test_data <- tibble::tibble(
  id = as.character(1:8),
  trauma_level = c("I", "II", "III", "IV", "I", "II", "III", "IV"),
  ed_disp = c(
    "Operating Room",
    "Admitted",
    "Deceased/Expired",
    "Transferred",
    "Deceased/Expired",
    "Deceased/Expired",
    "Admitted",
    "Deceased/Expired"
  ),
  ed_los = c(120, 200, 5000, 180, 3000, 4321, 60, 4000),
  hosp_disp = c(
    "Deceased/Expired",
    "Deceased/Expired",
    "Deceased/Expired",
    "Discharged",
    "Deceased/Expired",
    "Deceased/Expired",
    "Discharged",
    "Deceased/Expired"
  ),
  hosp_los = c(3000, 4500, 1000, 200, 5000, 4400, 150, 3000),
  autopsy_done = c("Yes", "No", "No", NA, "Yes", "No", NA, "Yes")
)

# Run the indicator function
traumar::seqic_indicator_4(
  data = test_data,
  level = trauma_level,
```

```
  ed_disposition = ed_disp,
  ed_LOS = ed_los,
  hospital_disposition = hosp_disp,
  hospital_LOS = hosp_los,
  unique_incident_id = id,
  autopsy = autopsy_done
)
```

`seqic_indicator_5` *SEQIC Indicator 5 - Alcohol and Drug Screening*

Description

[Experimental]

Computes SEQIC Indicator 5a–5d for trauma system quality monitoring. These indicators measure alcohol and drug screening rates among trauma patients at trauma level I–IV facilities.

Usage

```
seqic_indicator_5(
  data,
  level,
  included_levels = c("I", "II", "III", "IV"),
  unique_incident_id,
  blood_alcohol_content,
  drug_screen,
  groups = NULL,
  calculate_ci = NULL,
  ...
)
```

Arguments

`data`	A data frame containing trauma incident records.
`level`	Column indicating the trauma center designation level (e.g., I, II, III, IV).
`included_levels`	Character vector indicating what facility levels to include in the analysis. Defaults to `c("I", "II", "III", "IV")`.
`unique_incident_id`	Unique identifier for each record.
`blood_alcohol_content`	Unquoted column name for blood alcohol concentration. Numeric. A non-missing value indicates a test was performed. Values greater than zero are considered positive results.

`drug_screen` Unquoted column name for the drug screen result. Character or factor. May contain keywords (e.g., "opioid", "cocaine", "none"). The keywords used in this function correspond to the National Trauma Data Bank (NTDB) field values for the corresponding data element.

`groups` Additional columns passed as a vector of strings to `dplyr::summarize()` via the `.by` argument for grouped summaries. Defaults to `NULL`.

`calculate_ci` If `NULL`, 95% confidence intervals will not be calculated for the performance estimates. Otherwise, options of "wilson" or "clopper-pearson" can be supplied to utilize the corresponding methods to calculate the confidence intervals for the proportions. Defaults to `NULL`.

`...` Arguments passed on to `nemsqar::nemsqa_binomial_confint`

`conf.level` Numeric value between 0 and 1 indicating the confidence level. Defaults to 0.95 (95% confidence interval).

`correct` Logical, indicating whether to apply continuity correction for Wilson intervals. Defaults to `TRUE`.

Details

This function:

- Filters to trauma records at trauma levels I–IV.
- Deduplicates by `unique_incident_id` to ensure one record per incident.
- Calculates four sub-measures:
  - Indicator 5a: Proportion of patients with a blood alcohol test performed.
  - Indicator 5b: Among those tested, the proportion with BAC > 0.
  - Indicator 5c: Proportion of patients with any recorded drug screen result.
  - Indicator 5d: Among those with a drug result, the proportion that included a known positive drug (e.g., opioids, cocaine, THC).
- Matches drug-related terms using regular expressions for a broad set of known substances. Matching is case-insensitive.

Value

A tibble summarizing SEQIC Indicator 5a–5d results. Includes numerator, denominator, and calculated proportion for each measure. Optionally includes 95% confidence intervals.

Note

Users must ensure input columns are correctly named and contain standardized values where applicable. Drug screen values should ideally use consistent naming or be mapped to recognizable substance terms prior to function use.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Packages
library(dplyr)
library(traumar)

# Create synthetic test data for Indicators 5a–5d
test_data <- tibble::tibble(
  id = as.character(1:10),
  trauma_level = rep(c("I", "II", "III", "IV", "V"), each = 2),
  bac = c(0.08, NA, 0, 0.02, NA, 0.15, NA, NA, 0, 0),
  drug = c(
    "opioid", "none", "cocaine", "none", NA,
    "benzodiazepine", "alcohol", "thc", "none", NA
  )
)

# Run the indicator function
traumar::seqic_indicator_5(
  data = test_data,
  level = trauma_level,
  unique_incident_id = id,
  blood_alcohol_content = bac,
  drug_screen = drug
) |>
  tidyr::pivot_longer(cols = -1, names_to = "Indicator", values_to =
  "Values")
```

---

seqic_indicator_6 *SEQIC Indicator 6 - Delayed Arrival Following Low GCS*

Description

[Experimental]

Computes SEQIC Indicator 6 for trauma system quality monitoring. This indicator measures the proportion of patients presenting with a Glasgow Coma Scale (GCS) score < 9 who arrive at a trauma level I–IV center more than 180 minutes after injury. It excludes patients transferred out of the facility and focuses on those transferred into a facility.

Usage

```
seqic_indicator_6(
  data,
  level,
  included_levels = c("I", "II", "III", "IV"),
  unique_incident_id,
  transfer_out_indicator,
  receiving_indicator,
```

```
  low_GCS_indicator,
  time_from_injury_to_arrival,
  groups = NULL,
  calculate_ci = NULL,
  ...
)
```

Arguments

`data`	A data frame containing trauma incident records.
`level`	Column indicating the trauma center designation level (e.g., I, II, III, IV).
`included_levels`	Character vector indicating what facility levels to include in the analysis. Defaults to `c("I", "II", "III", "IV")`.
`unique_incident_id`	Unique identifier for each record.
`transfer_out_indicator`	Column name indicating whether the patient was transferred out of the initial trauma center to definitive care. Logical, character, or factor type. Values representing "No" (e.g., FALSE, "No") indicate no transfer out.
`receiving_indicator`	Column name indicating whether the patient was transferred into the trauma center. Logical, character, or factor type. Values representing "Yes" (e.g., TRUE, "Yes") indicate transfer in.
`low_GCS_indicator`	Column name for identifying patients with a Glasgow Coma Scale score less than 9. Logical, character, or factor type.
`time_from_injury_to_arrival`	Column name representing the time in minutes from injury occurrence to arrival at the trauma center. Numeric type.
`groups`	Additional columns passed as a vector of strings to `dplyr::summarize()` via the `.by` argument for grouped summaries. Defaults to `NULL`.
`calculate_ci`	If `NULL`, 95% confidence intervals will not be calculated for the performance estimates. Otherwise, options of "wilson" or "clopper-pearson" can be supplied to utilize the corresponding methods to calculate the confidence intervals for the proportions. Defaults to `NULL`.
`...`	Arguments passed on to `nemsqar::nemsqa_binomial_confint` `conf.level` Numeric value between 0 and 1 indicating the confidence level. Defaults to 0.95 (95% confidence interval). `correct` Logical, indicating whether to apply continuity correction for Wilson intervals. Defaults to `TRUE`.

Details

This function:

- Filters to trauma center records from facilities at trauma levels I–IV.

- Deduplicates records using `unique_incident_id`.
- Calculates:
  - Numerator: Patients with low GCS (< 9) who arrived more than 180 minutes after injury, were transferred in, and not transferred out.
  - Denominator: All patients with low GCS (< 9) who were transferred in and not transferred out.
- Optionally calculates Wilson or Clopper-Pearson confidence intervals for the resulting proportion if `calculate_ci` is specified.

Value

A tibble summarizing SEQIC Indicator 6 results. Includes numerator, denominator, calculated proportion, and optionally 95% confidence intervals.

Note

Users must ensure input columns are appropriately coded and standardized. Transfer and GCS indicators should use consistent logical or textual representations.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Packages
library(dplyr)
library(traumar)

# Create test data for Indicator 6
test_data <- tibble::tibble(
  id = as.character(1:10),
  trauma_level = rep(c("I", "II", "III", "IV", "V"), times = 2),
  transfer_out = c("No", "No", "Yes", "No", "No", "No", "No", "No", "No",
  "No"),
  transfer_in = c("Yes", "Yes", "No", "Yes", "No", "Yes", "Yes", "Yes",
  "Yes", "Yes"),
  gcs_low = c(TRUE, TRUE, TRUE, FALSE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE),
  time_to_arrival = c(200, 100, 300, 190, 400, 181, 100, 179, 240, 178)
)

# Run the indicator function
traumar::seqic_indicator_6(
  data = test_data,
  level = trauma_level,
  unique_incident_id = id,
  transfer_out_indicator = transfer_out,
  receiving_indicator = transfer_in,
  low_GCS_indicator = gcs_low,
  time_from_injury_to_arrival = time_to_arrival
```

```
)
```

`seqic_indicator_7` *SEQIC Indicator 7 - Delayed Arrival to Definitive Care*

Description

[Experimental]

Computes SEQIC Indicator 7, which measures the proportion of trauma patients arriving at the definitive care facility trauma centers (level I–IV) more than 180 minutes after injury. This indicator identifies delays in definitive care.

Usage

```
seqic_indicator_7(
  data,
  level,
  included_levels = c("I", "II", "III", "IV"),
  unique_incident_id,
  time_from_injury_to_arrival,
  transfer_out_indicator,
  groups = NULL,
  calculate_ci = NULL,
  ...
)
```

Arguments

`data`	A data frame containing trauma incident records.
`level`	Column indicating the trauma center designation level (e.g., I, II, III, IV).
`included_levels`	Character vector indicating what facility levels to include in the analysis. Defaults to `c("I", "II", "III", "IV")`.
`unique_incident_id`	Unique identifier for each record.
`time_from_injury_to_arrival`	Column name representing the time in minutes from injury occurrence to arrival at the trauma center. Numeric type.
`transfer_out_indicator`	Column name indicating whether the patient was transferred out of the initial trauma center to definitive care. Logical, character, or factor type. Values representing "No" (e.g., FALSE, "No") indicate no transfer out.
`groups`	Additional columns passed as a vector of strings to `dplyr::summarize()` via the `.by` argument for grouped summaries. Defaults to `NULL`.

`calculate_ci` If `NULL`, 95% confidence intervals will not be calculated for the performance estimates. Otherwise, options of "wilson" or "clopper-pearson" can be supplied to utilize the corresponding methods to calculate the confidence intervals for the proportions. Defaults to `NULL`.

`...` Arguments passed on to `nemsqar::nemsqa_binomial_confint`

`conf.level` Numeric value between 0 and 1 indicating the confidence level. Defaults to 0.95 (95% confidence interval).

`correct` Logical, indicating whether to apply continuity correction for Wilson intervals. Defaults to `TRUE`.

Details

This function:

- Filters the dataset to trauma center levels I through IV.
- Deduplicates the dataset by `unique_incident_id`.
- Creates a logical flag for arrivals occurring more than 180 minutes after injury.
- Identifies definitive care records where the patient arrived greater than 180 minutes after the time of injury.
- Returns a summarized tibble with the number of such cases (numerator), total eligible records (denominator), and the proportion.
- Optionally includes 95% confidence intervals if `calculate_ci` is specified.

Value

A tibble summarizing SEQIC Indicator 7 results. Includes numerator, denominator, and proportion. 95% confidence intervals are included if requested.

Note

The user must ensure all columns are correctly passed and that time values are numeric and measured in minutes.

Author(s)

Nicolas Foss Ed.D., MS

Examples

```
# Packages
library(dplyr)
library(traumar)

# Create test data for Indicator 7
test_data <- tibble::tibble(
  id = as.character(1:10),
  trauma_level = rep(c("I", "II", "III", "IV", "V"), times = 2),
  time_to_arrival = c(200, 100, 220, 150, 400, 181, 90, 179, 240, 178),
```

```
  transfer_out = c("No", "No", "No", "No", "Yes", "No", "No", "No", "No",
  "No")
)

# Run the indicator function
traumar::seqic_indicator_7(
  data = test_data,
  level = trauma_level,
  unique_incident_id = id,
  time_from_injury_to_arrival = time_to_arrival,
  transfer_out_indicator = transfer_out
)
```

`seqic_indicator_8` *SEQIC Indicator 8 - Survival by Risk Group*

Description

[Experimental]

Calculates the proportion of patients who survived based on risk groups existing in the data among trauma patients transported to Level I–IV trauma centers.

Usage

```
seqic_indicator_8(
  data,
  level,
  included_levels = c("I", "II", "III", "IV"),
  unique_incident_id,
  mortality_indicator,
  risk_group,
  groups = NULL,
  calculate_ci = NULL,
  ...
)
```

Arguments

`data`	A data frame containing trauma incident records.
`level`	Column indicating the trauma center designation level (e.g., I, II, III, IV).
`included_levels`	Character vector indicating what facility levels to include in the analysis. Defaults to `c("I", "II", "III", "IV")`.
`unique_incident_id`	Unique identifier for each record.

`mortality_indicator`
: A logical, character, or factor variable indicating whether the patient died at the trauma center. Accepts values like `TRUE`/`FALSE` or `"Yes"`/`"No"`.

`risk_group`
: A character or factor column indicating the patient's risk group (e.g., "High", "Moderate", "Low"). See risk definitions below.

`groups`
: Additional columns passed as a vector of strings to `dplyr::summarize()` via the `.by` argument for grouped summaries. Defaults to `NULL`.

`calculate_ci`
: If `NULL`, 95% confidence intervals will not be calculated for the performance estimates. Otherwise, options of "wilson" or "clopper-pearson" can be supplied to utilize the corresponding methods to calculate the confidence intervals for the proportions. Defaults to `NULL`.

`...`
: Arguments passed on to `nemsqar::nemsqa_binomial_confint`

  `conf.level` Numeric value between 0 and 1 indicating the confidence level. Defaults to 0.95 (95% confidence interval).

  `correct` Logical, indicating whether to apply continuity correction for Wilson intervals. Defaults to `TRUE`.

Details

- Filters the dataset to include only trauma center levels I through IV.
- Deduplicates the dataset using `unique_incident_id` to ensure one record per incident.
- Accepts a mortality indicator that may be logical, character, or factor, and identifies survivors as those with values of `FALSE` or `"No"`.
- Requires a predefined `risk_group` variable representing categories such as "Low", "Moderate", or "High" risk.
- Calculates overall survival proportions and survival proportions stratified by risk group.
- Optionally includes 95% confidence intervals using binomial methods if `calculate_ci` is specified.

Value

A named list with two tibbles:

`overall`: A tibble summarizing overall mortality among trauma patients, grouped by the variables specified in `groups`. Columns include:

- `numerator_8_all` (number of survivors),
- `denominator_8_all` (total number of unique trauma incidents),
- `seqic_8_all` (survival proportion), and optionally
- `lower_ci_8`,
- `upper_ci_8` (confidence interval bounds if `calculate_ci` is specified).

`risk_group`: A tibble summarizing mortality stratified by risk group and any additional grouping variables. Columns include:

- `risk_group` (used for stratification),

- `numerator_8_risk` (survivors per group),
- `denominator_8_risk` (total incidents per group),
- `seqic_8_risk` (survival proportion per group), and optionally
- `lower_ci_8_risk`,
- `upper_ci_8_risk` (confidence interval bounds if `calculate_ci` is specified).

Note

This function calculates survival outcomes for patients transported to trauma centers, stratified by risk of mortality. Risk groups—low, moderate, and high— are defined by the Iowa System Evaluation and Quality Improvement Committee (SEQIC) as described below. Users may also apply alternative risk stratification methods if preferred.

- Abnormal Physiology Criteria: GCS 3–5; Respirations <5 or >30 per minute; Systolic BP <60 mm Hg
- High Risk: Probability of Survival < 0.2; ISS > 41; ISS > 24 with abnormal physiology
- Moderate Risk: Probability of Survival 0.2–0.5; ISS 16–41
- Low Risk: Probability of Survival > 0.5; ISS < 16; Normal physiology

Users must ensure appropriate column names are passed and data is pre-processed to include the necessary fields without missing critical identifiers or timestamps.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Packages
library(dplyr)
library(traumar)

# Simulated dataset for SEQIC Indicator 8
test_data <- tibble::tibble(
  id = as.character(1:12),
  trauma_level = c("I", "II", "III", "IV", "V", "II", "I", "III", "IV", "II",
  "I", "III"),
  mortality = c(FALSE, "No", TRUE, "Yes", FALSE, TRUE, "No", FALSE, "Yes",
  FALSE, TRUE, "No"),
  risk = c("High", "High", "Moderate", "Moderate", "Low", "Low", "High",
  "Moderate", "Low", "Moderate", "High", "Low")
)

# Run indicator 8 function
traumar::seqic_indicator_8(
  data = test_data,
  level = trauma_level,
  unique_incident_id = id,
  mortality_indicator = mortality,
```

```
  risk_group = risk
)
```

`seqic_indicator_9` *SEQIC Indicator 9 - Emergency Department Transfer Timeliness*

Description

[Experimental]

Calculates the proportion of EMS-transferred trauma patients who experienced delayed transfer from the emergency department (ED) based on disposition and decision-to-transfer time frames. This includes both overall rates and stratified results by trauma team activation status, with optional confidence intervals.

Usage

```
seqic_indicator_9(
  data,
  level,
  included_levels = c("I", "II", "III", "IV"),
  transfer_out_indicator,
  transport_method,
  unique_incident_id,
  trauma_team_activated,
  risk_group,
  ed_LOS,
  ed_decision_LOS,
  ed_decision_discharge_LOS,
  groups = NULL,
  calculate_ci = NULL,
  ...
)
```

Arguments

`data`	A data frame containing trauma incident records.
`level`	Column indicating the trauma center designation level (e.g., I, II, III, IV).
`included_levels`	Character vector indicating what facility levels to include in the analysis. Defaults to `c("I", "II", "III", "IV")`.
`transfer_out_indicator`	Column name indicating whether the patient was transferred out of the initial trauma center to definitive care. Logical, character, or factor type. Values representing "No" (e.g., FALSE, "No") indicate no transfer out.

`transport_method`
: Column identifying the EMS transport method (e.g., ambulance, private vehicle). Used to exclude non-qualified modes of arrival.

`unique_incident_id`
: Unique identifier for each record.

`trauma_team_activated`
: Column indicating whether the trauma team was activated (character, factor, or logical).

`risk_group`
: A character or factor column indicating the patient's risk group (e.g., "High", "Moderate", "Low"). See risk definitions below.

`ed_LOS`
: Column for the calculated ED length of stay, measured in minutes.

`ed_decision_LOS`
: Numeric column representing minutes from ED arrival to decision to transfer.

`ed_decision_discharge_LOS`
: Numeric column representing minutes from ED decision to discharge to physical discharge.

`groups`
: Additional columns passed as a vector of strings to `dplyr::summarize()` via the `.by` argument for grouped summaries. Defaults to `NULL`.

`calculate_ci`
: If `NULL`, 95% confidence intervals will not be calculated for the performance estimates. Otherwise, options of "wilson" or "clopper-pearson" can be supplied to utilize the corresponding methods to calculate the confidence intervals for the proportions. Defaults to `NULL`.

`...`
: Arguments passed on to `nemsqar::nemsqa_binomial_confint`

  `conf.level` Numeric value between 0 and 1 indicating the confidence level. Defaults to 0.95 (95% confidence interval).

  `correct` Logical, indicating whether to apply continuity correction for Wilson intervals. Defaults to `TRUE`.

Details

This function:

- Filters the dataset to include only transfers out from trauma centers designated Level I through IV.
- Deduplicates records using `unique_incident_id`.
- Flags records where emergency department decision to discharge occurred more than 60 or 120 minutes after ED arrival.
- Flags records where physical departure from the ED occurred more than 120 or 180 minutes after ED arrival.
- Flags records where physical discharge occurred more than 60 or 120 minutes after ED decision to discharge.
- Stratifies results by trauma team activation status and one or more grouping variables.
- Stratifies results by risk groups and one or more grouping variables.
- Returns a summarized tibble with the number of delayed cases (numerator), eligible records (denominator), and the proportion for each delay threshold.
- Optionally includes 95% confidence intervals if `calculate_ci = TRUE`.

Value

A list of four tibbles, with optional 95% confidence intervals:

- `seqic_9_all`: Proportion of transferred trauma patients with ED discharge or decision delays >2 or >3 hours, grouped by optional variables.
- `seqic_9_activations`: Same proportions as above, further stratified by trauma team activation status.
- `seqic_9_risk`: Same proportions as above, further stratified by risk groups.
- `seqic_9_activations_risk`: Same proportions as above, further stratified by risk groups and trauma team activation status.

Each tibble includes numerators, denominators, proportions, and (optionally) confidence intervals for:

- 9a: Delayed discharge >2 hours
- 9b: Delayed discharge >3 hours
- 9c: Delayed decision >1 hours
- 9d: Delayed decision >2 hours
- 9e: Delayed decision to discharge >1 hour
- 9f: Delayed decision to discharge >2 hours

Note

This function calculates discharge timeliness outcomes for patients transported to trauma centers, stratified by risk of mortality. Risk groups—low, moderate, and high— are defined by the Iowa System Evaluation and Quality Improvement Committee (SEQIC) as described below. Users may also apply alternative risk stratification methods if preferred.

- Abnormal Physiology Criteria: GCS 3–5; Respirations <5 or >30 per minute; Systolic BP <60 mm Hg
- High Risk: Probability of Survival < 0.2; ISS > 41; ISS > 24 with abnormal physiology
- Moderate Risk: Probability of Survival 0.2–0.5; ISS 16–41
- Low Risk: Probability of Survival > 0.5; ISS < 16; Normal physiology

Users must ensure appropriate column names are passed and data is pre-processed to include the necessary fields without missing critical identifiers or timestamps.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Packages
library(dplyr)
library(traumar)

# Simulated dataset for SEQIC Indicator 9
test_data <- tibble::tibble(
  id = as.character(1:10),
  trauma_level = c("I", "II", "III", "IV", "V", "II", "III", "IV", "I",
  "II"),
  transport = c("Ambulance", "Ambulance", "Private Vehicle", "Ambulance",
  "Helicopter", "Ambulance", "Ambulance", "Ambulance", "Ambulance",
  "Ambulance"),
  activated = c(TRUE, FALSE, TRUE, FALSE, TRUE, FALSE, TRUE, TRUE, FALSE,
  FALSE),
  ed_LOS = c(120, 180, 90, 60, 200, 130, 110, 160, 95, 220),
  ed_decision = c(55, 125, 65, 30, 190, 80, 70, 45, 61, 130),
  ed_discharge = c(130, 185, 110, 65, 150, 160, 95, 180, 70, 210),
  transfer_out = c(TRUE, TRUE, FALSE, TRUE, TRUE, TRUE, TRUE, FALSE, TRUE,
  TRUE),
  risk = c("High", "High", "Moderate", "Low", "Moderate", "Low",
           "High", "Low", "Moderate", "High")
)

# Run the function, and store as a list object
seqic_9_result <- traumar::seqic_indicator_9(
  data = test_data,
  level = trauma_level,
  included_levels = c("I", "II", "III", "IV"),
  unique_incident_id = id,
  transport_method = transport,
  transfer_out_indicator = transfer_out,
  ed_LOS = ed_LOS,
  ed_decision_LOS = ed_decision,
  ed_decision_discharge_LOS = ed_discharge,
  trauma_team_activated = activated,
  risk_group = risk
)

# Take a look at the overall output of the function
seqic_9_result$overall |>
tidyr::pivot_longer(cols = -1,
                    names_to = "Indicator",
                    values_to = "Values"
                    )
```

---

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 values representing a binary outcome. It also ensures that Ps_col contains numeric values within the range 0 to 1. If any values exceed 1, a warning is issued. 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.

Like other statistical computing functions, trauma_case_mix() is happiest without missing data. It is best to pass complete probability of survival and outcome data to the function for optimal performance. With smaller datasets, this is especially helpful. However, trauma_case_mix() will throw a warning about missing values, if any exist in Ps_col and/or outcome_col.

Value

A data frame containing:

- Ps_range: The probability of survival range category.
- current_fraction: The fraction of patients in the current dataset within each Ps range.
- MTOS_distribution: The reference distribution of patients in each Ps range based on the MTOS study.
- survivals: The number of observed survivors (outcome = 1) in each Ps range.
- predicted_survivals: The sum of predicted survivals (sum of Ps values) in each Ps range.
- deaths: The number of observed deaths (outcome = 0) in each Ps range.
- predicted_deaths: The sum of predicted deaths (sum of 1 - Ps values) in each Ps range.
- count: The total number of patients in each Ps range.

Note

This function will produce the most reliable and interpretable results when using a dataset that has one row per patient, with each column being a feature.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Generate example data
set.seed(123)

# Parameters
# Total number of patients
n_patients <- 5000

# Arbitrary group labels
groups <- sample(x = LETTERS[1:2], size = n_patients, replace = TRUE)

# Trauma types
trauma_type_values <- sample(
  x = c("Blunt", "Penetrating"),
  size = n_patients,
  replace = TRUE
)

# RTS values
rts_values <- sample(
  x = seq(from = 0, to = 7.8408, by = 0.005),
  size = n_patients,
  replace = TRUE
)

# patient ages
ages <- sample(
  x = seq(from = 0, to = 100, by = 1),
  size = n_patients,
  replace = TRUE
)

# ISS scores
iss_scores <- sample(
  x = seq(from = 0, to = 75, by = 1),
  size = n_patients,
  replace = TRUE
)

# Generate survival probabilities (Ps)
Ps <- traumar::probability_of_survival(
  trauma_type = trauma_type_values,
  age = ages,
  rts = rts_values,
  iss = iss_scores
)

# 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, groups = groups) |>
```

```
  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,
  z_method = c("survival", "mortality")
)
```

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). The function will check values in this column and expects them to represent the outcome in a binary form.
`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".

Details

The function checks whether the `outcome_col` contains values representing a binary outcome. It also ensures that `Ps_col` contains numeric values within the range 0 to 1. If any values exceed 1, a warning is issued. 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.

Like other statistical computing functions, `trauma_performance()` is happiest without missing data. It is best to pass complete probability of survival and outcome data to the function for optimal performance. With smaller datasets, this is especially helpful. However, `trauma_performance()` will throw a warning about missing values, if any exist in `Ps_col` and/or `outcome_col`.

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 that were predicted to die, calculated based on the W-score. This value reflects the difference between the actual and predicted number of deceased patients.
- `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.

Note

This function will produce the most reliable and interpretable results when using a dataset that has one row per patient, with each column being a feature.

Author(s)

Nicolas Foss, Ed.D., MS

Examples

```
# Generate example data
set.seed(123)

# Parameters
```

```
# Total number of patients
n_patients <- 5000

# Arbitrary group labels
groups <- sample(x = LETTERS[1:2], size = n_patients, replace = TRUE)

# Trauma types
trauma_type_values <- sample(
  x = c("Blunt", "Penetrating"),
  size = n_patients,
  replace = TRUE
)

# RTS values
rts_values <- sample(
  x = seq(from = 0, to = 7.8408, by = 0.005),
  size = n_patients,
  replace = TRUE
)

# patient ages
ages <- sample(
  x = seq(from = 0, to = 100, by = 1),
  size = n_patients,
  replace = TRUE
)

# ISS scores
iss_scores <- sample(
  x = seq(from = 0, to = 75, by = 1),
  size = n_patients,
  replace = TRUE
)

# Generate survival probabilities (Ps)
Ps <- traumar::probability_of_survival(
  trauma_type = trauma_type_values,
  age = ages,
  rts = rts_values,
  iss = iss_scores
)

# 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, groups = groups) |>
  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)
```

validate_character_factor *Validate a Character or Factor Input*

Description

This function checks if an input is of type character or factor. Depending on the specified type, it will either throw an error, issue a warning, or send a message. It also checks for NULL and NA values based on the specified parameters.

Usage

```
validate_character_factor(
  input,
  type = c("error", "warning", "message"),
  na_ok = TRUE,
  null_ok = TRUE,
  var_name = NULL,
  calls = NULL
)
```

Arguments

input	The data to be validated.
type	A character string specifying the type of message to be displayed if the input is not numeric or if the values are out of range. Must be one of "error", "warning", or "message".
na_ok	Logical. If TRUE, NA values are allowed. Default is TRUE.
null_ok	Logical. If TRUE, NULL values are allowed. Default is TRUE.
var_name	Optional. A character string giving the desired variable (or object) name that will appear in console output in place of the how the object will typically be named in messages via deparse(substitute(input)).
calls	Optional. The number of callers to go back in the call stack for error messaging. If NULL, will default to 2.

Value

NULL. The function is used for its side effects.

Author(s)

Nicolas Foss, Ed.D., MS

validate_choice *Validate Choice*

Description

This function checks if an input is within a specified set of valid choices. Depending on the specified type, it will either throw an error, issue a warning, or send a message.

Usage

```
validate_choice(
  input,
  choices,
  several.ok = FALSE,
  type = c("error", "warning", "message"),
  na_ok = TRUE,
  null_ok = TRUE,
  var_name = NULL,
  calls = NULL
)
```

Arguments

input	The data to be validated.
choices	a character vector of candidate values, often missing, see documentation for base::match.arg() for more information.
several.ok	logical specifying if arg should be allowed to have more than one element.
type	A character string specifying the type of message to be displayed if the input is not numeric or if the values are out of range. Must be one of "error", "warning", or "message".
na_ok	Logical. If TRUE, NA values are allowed. Default is TRUE.
null_ok	Logical. If TRUE, NULL values are allowed. Default is TRUE.
var_name	Optional. A character string giving the desired variable (or object) name that will appear in console output in place of the how the object will typically be named in messages via deparse(substitute(input)).
calls	Optional. The number of callers to go back in the call stack for error messaging. If NULL, will default to 2.

Details

This function uses match.arg to validate the input against the allowed choices. Please see the documentation for match.arg for more details about how matching is performed.

Value

The validated input if it is valid.

Author(s)

Nicolas Foss, Ed.D., MS

`validate_class` *Validate Class*

Description

This function checks if the input is of the specified class type(s). Depending on the specified type, it will either throw an error, issue a warning, or send a message. It also checks for NULL and NA values based on the specified parameters.

Usage

```
validate_class(
  input,
 class_type = c("numeric", "integer", "logical", "character", "factor", "complex",
    "raw", "date", "date-time", "hms"),
  logic = c("or", "and"),
  type = c("error", "warning", "message"),
  na_ok = TRUE,
  null_ok = TRUE,
  finite = FALSE,
  var_name = NULL,
  calls = NULL
)
```

Arguments

`input`	The data to be validated.
`class_type`	A vector of class types to check. Possible values are "numeric", "integer", "logical", "character", "factor", "complex", "raw", "date", "date-time", "hms".
`logic`	The logical operator to use when combining checks. Possible values are "or", and "and".
`type`	A character string specifying the type of message to be displayed if the input is not numeric or if the values are out of range. Must be one of "error", "warning", or "message".
`na_ok`	Logical. If TRUE, NA values are allowed. Default is TRUE.
`null_ok`	Logical. If TRUE, NULL values are allowed. Default is TRUE.
`finite`	Logical. If TRUE, only finite values are allowed. Default is FALSE.
`var_name`	Optional. A character string giving the desired variable (or object) name that will appear in console output in place of the how the object will typically be named in messages via deparse(substitute(input)).
`calls`	Optional. The number of callers to go back in the call stack for error messaging. If NULL, will default to 2.

Value

NULL. The function is used for its side effects.

Author(s)

Nicolas Foss, Ed.D., MS

`validate_complete` *Validate Complete Input*

Description

This function checks if the input contains any missing values (NA). Depending on the specified type, it will either throw an error, issue a warning, or send a message. It also checks for NULL values based on the specified parameters.

Usage

```
validate_complete(
  input,
  type = c("error", "warning", "message"),
  null_ok = TRUE,
  var_name = NULL,
  calls = NULL
)
```

Arguments

`input`	The data to be validated.
`type`	A character string specifying the type of message to be displayed if the input is not numeric or if the values are out of range. Must be one of "error", "warning", or "message".
`null_ok`	Logical. If TRUE, NULL values are allowed. Default is TRUE.
`var_name`	Optional. A character string giving the desired variable (or object) name that will appear in console output in place of the how the object will typically be named in messages via deparse(substitute(input)).
`calls`	Optional. The number of callers to go back in the call stack for error messaging. If NULL, will default to 2.

Value

NULL. The function is used for its side effects.

Author(s)

Nicolas Foss, Ed.D., MS

`validate_data_pull` *Validate Data Extraction*

Description

This function extracts a column from a data frame or tibble and returns it as a vector. If the column does not exist or an error occurs, it returns a clean error message using the cli package.

Usage

```
validate_data_pull(
  input,
  type = c("error", "warning", "message"),
  col,
  var_name = NULL,
  calls = NULL
)
```

Arguments

`input`	A data frame or tibble.
`type`	A character string specifying the type of message to be displayed if the input is not numeric or if the values are out of range. Must be one of "error", "warning", or "message".
`col`	The column to be extracted.
`var_name`	Optional. The name of the variable for error messaging.
`calls`	Optional. The number of callers to go back in the call stack for error messaging. If NULL, will default to 2.

Details

This function is designed to validate and extract a specified column from a data frame or tibble. When using `validate_data_pull()` within custom functions, it is necessary to call a given bare column name using tidy evaluation (e.g., `{{ col }}`). This allows the function to correctly capture and evaluate the column name within the custom function. However, when calling this function directly on a data frame, tidy evaluation is not required.

Value

The extracted column as a vector.

Author(s)

Nicolas Foss, Ed.D., MS

validate_data_structure *Validate Data Structure*

Description

This function checks if an input is of the specified data structure type(s). Depending on the specified type, it will either throw an error, issue a warning, or send a message. It also checks for NULL and NA values based on the specified parameters.

Usage

```
validate_data_structure(
  input,
  structure_type = c("data.frame", "matrix", "list", "array", "atomic_vector", "tbl_df",
    "tbl"),
  logic = c("or", "and"),
  type = c("error", "warning", "message"),
  na_ok = TRUE,
  null_ok = TRUE,
  var_name = NULL,
  calls = NULL
)
```

Arguments

input	The data to be validated.
structure_type	A vector of data structure types to check. Possible values are "data.frame", "matrix", "list", "array", "atomic_vector", "tbl_df", "tbl".
logic	The logical operator to use when combining checks. Possible values are "or", and "and".
type	A character string specifying the type of message to be displayed if the input is not numeric or if the values are out of range. Must be one of "error", "warning", or "message".
na_ok	Logical. If TRUE, NA values are allowed. Default is TRUE.
null_ok	Logical. If TRUE, NULL values are allowed. Default is TRUE.
var_name	Optional. A character string giving the desired variable (or object) name that will appear in console output in place of the how the object will typically be named in messages via deparse(substitute(input)).
calls	Optional. The number of callers to go back in the call stack for error messaging. If NULL, will default to 2.

Value

NULL. The function is used for its side effects.

Author(s)

Nicolas Foss, Ed.D., MS

`validate_error_type` *Validate Error Type*

Description

This function displays an error, warning, or message based on the specified type.

Usage

```
validate_error_type(
  input,
  message,
  type = c("error", "warning", "message"),
  calls = NULL
)
```

Arguments

`input`	The data to be validated.
`message`	The message to be displayed.
`type`	A character string specifying the type of message to be displayed if the input is not numeric or if the values are out of range. Must be one of "error", "warning", or "message".
`calls`	Optional. The number of callers to go back in the call stack for error messaging. If NULL, will default to 2.

Value

NULL. The function is used for its side effects.

`validate_length` *Validate Length of an Input*

Description

This function checks if the length of a vector or list is within a specified range. Depending on the specified type, it will either throw an error, issue a warning, or send a message. It also checks for NULL and NA values based on the specified parameters.

Usage

```
validate_length(
  input,
  exact_length = NULL,
  min_length = NULL,
  max_length = NULL,
  type = c("error", "warning", "message"),
  na_ok = TRUE,
  null_ok = TRUE,
  var_name = NULL,
  calls = NULL
)
```

Arguments

input	The data to be validated.
exact_length	The required length of the vector or list. If this argument is used, then min_length and max_length are not required.
min_length	The minimum length of the vector or list.
max_length	The maximum length of the vector or list.
type	A character string specifying the type of message to be displayed if the input is not numeric or if the values are out of range. Must be one of "error", "warning", or "message".
na_ok	Logical. If TRUE, NA values are allowed. Default is TRUE.
null_ok	Logical. If TRUE, NULL values are allowed. Default is TRUE.
var_name	Optional. A character string giving the desired variable (or object) name that will appear in console output in place of the how the object will typically be named in messages via deparse(substitute(input)).
calls	Optional. The number of callers to go back in the call stack for error messaging. If NULL, will default to 2.

Value

NULL. The function is used for its side effects.

Author(s)

Nicolas Foss, Ed.D., MS

validate_names *Validate Column Names*

Description

This function checks if all column names of a data frame or tibble are within a specified set of valid values. Depending on the specified type, it will either throw an error, issue a warning, or send a message.

Usage

```
validate_names(
  input,
  check_names,
  type = c("error", "warning", "message"),
  na_ok = TRUE,
  null_ok = TRUE,
  var_name = NULL,
  calls = NULL
)
```

Arguments

input	A data.frame or tibble. validate_names() will run colnames(input) to get the expected column names.
check_names	A vector of column names as strings to check against input.
type	A character string specifying the type of message to be displayed if the input is not numeric or if the values are out of range. Must be one of "error", "warning", or "message".
na_ok	Logical. If TRUE, NA values are allowed. Default is TRUE.
null_ok	Logical. If TRUE, NULL values are allowed. Default is TRUE.
var_name	Optional. A character string giving the desired variable (or object) name that will appear in console output in place of the how the object will typically be named in messages via deparse(substitute(input)).
calls	Optional. The number of callers to go back in the call stack for error messaging. If NULL, will default to 2.

Value

NULL. The function is used for its side effects.

Author(s)

Nicolas Foss, Ed.D., MS

validate_numeric *Validate Numeric Input*

Description

This function checks if an input is numeric and optionally checks if the values are within a specified range. Depending on the specified type, it will either throw an error, issue a warning, or send a message. Additional arguments allow for checking NA values, NULL values, and finite values.

Usage

```
validate_numeric(
  input,
  min = NULL,
  max = NULL,
  na_ok = TRUE,
  null_ok = TRUE,
  finite = FALSE,
  type = c("error", "warning", "message"),
  var_name = NULL,
  calls = NULL
)
```

Arguments

input	The data to be validated.
min	Optional. The minimum value for the range check.
max	Optional. The maximum value for the range check.
na_ok	Logical. If TRUE, NA values are allowed. Default is TRUE.
null_ok	Logical. If TRUE, NULL values are allowed. Default is TRUE.
finite	Logical. If TRUE, only finite values are allowed. Default is FALSE.
type	A character string specifying the type of message to be displayed if the input is not numeric or if the values are out of range. Must be one of "error", "warning", or "message".
var_name	Optional. A character string giving the desired variable (or object) name that will appear in console output in place of the how the object will typically be named in messages via deparse(substitute(input)).
calls	Optional. The number of callers to go back in the call stack for error messaging. If NULL, will default to 2.

Value

NULL. The function is used for its side effects.

Author(s)

Nicolas Foss, Ed.D., MS

`validate_set` *Validate Set Equality*

Description

This function checks if all elements of an input are within a specified set of valid values. Depending on the specified type, it will either throw an error, issue a warning, or send a message.

Usage

```
validate_set(
  input,
  valid_set,
  type = c("error", "warning", "message"),
  na_ok = TRUE,
  null_ok = TRUE,
  var_name = NULL,
  calls = NULL
)
```

Arguments

`input`	The data to be validated.
`valid_set`	A vector of valid values.
`type`	A character string specifying the type of message to be displayed if the input is not numeric or if the values are out of range. Must be one of "error", "warning", or "message".
`na_ok`	Logical. If TRUE, NA values are allowed. Default is TRUE.
`null_ok`	Logical. If TRUE, NULL values are allowed. Default is TRUE.
`var_name`	Optional. A character string giving the desired variable (or object) name that will appear in console output in place of the how the object will typically be named in messages via deparse(substitute(input)).
`calls`	Optional. The number of callers to go back in the call stack for error messaging. If NULL, will default to 2.

Value

NULL. The function is used for its side effects.

Author(s)

Nicolas Foss, Ed.D., MS

`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