Package 'tidycomm'

August 27, 2025

Title Data Modification and Analysis for Communication Research

Version 0.4.2

Description Provides convenience functions for common data modification and analysis tasks in communication research. This includes functions for univariate and bivariate data analysis, index generation and reliability computation, and intercoder reliability tests. All functions follow the style and syntax of the tidyverse, and are construed to perform their computations on multiple variables at once. Functions for univariate and bivariate data analysis comprise summary statistics for continuous and categorical variables, as well as several tests of bivariate association including effect sizes. Functions for data modification comprise index generation and automated reliability analysis of index variables. Functions for intercoder reliability comprise tests of several intercoder reliability estimates, including simple and mean pairwise percent agreement, Krippendorff's Alpha (Krippendorff 2004, ISBN: 9780761915454), and various Kappa coefficients (Brennan & Prediger 1981 <doi:10.1177/001316448104100307>; Cohen 1960 <doi:10.1177/001316446002000104>; Fleiss 1971 <doi:10.1037/h0031619>).

License GPL-3

URL `https://tidycomm.github.io/tidycomm/`

BugReports `https://github.com/tidycomm/tidycomm/issues`

Depends R (>= 3.6.0)

Imports car, dplyr, fastDummies, forcats, GGally, ggplot2, glue, lm.beta, lubridate, magrittr, MASS, MBESS, misty, pillar, purrr, rlang, stringr, tibble, tidyr, tidyselect

Suggests covr, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

Author Julian Unkel [aut, cre] (ORCID:
<https://orcid.org/0000-0001-9568-7041>),
Mario Haim [aut] (ORCID: <https://orcid.org/0000-0002-0643-2299>),
Lara Kobilke [aut] (ORCID: <https://orcid.org/0000-0001-6194-4724>)

Maintainer Julian Unkel <julian.unkel@gmail.com>

Repository CRAN

Date/Publication 2025-08-27 12:00:02 UTC

Contents

`add_index` *Add index*

Description

Add a rowwise mean or sum index of specific variables to the dataset.

Usage

```
add_index(data, name, ..., type = "mean", na.rm = TRUE, cast.numeric = FALSE)
```

Arguments

`data`	a tibble or a tdcmm model
`name`	Name of the index column to compute.
`...`	Variables used for the index.
`type`	Type of index to compute. Either "mean" (default) or "sum".
`na.rm`	a logical value indicating whether `NA` values should be stripped before the computation proceeds. Defaults to `TRUE`.
`cast.numeric`	a logical value indicating whether all variables selected for index computation should be converted to numeric. Useful if computing indices from factor variables. Defaults to `FALSE`.

Value

a tdcmm model

See Also

`get_reliability()` to compute reliability estimates of added index variables.

Examples

```
WoJ %>% add_index(ethical_flexibility, ethics_1, ethics_2, ethics_3, ethics_4)
WoJ %>% add_index(ethical_flexibility, ethics_1, ethics_2, ethics_3, ethics_4, type = "sum")
```

`categorize_scale` *Categorize numeric variables into categories*

Description

This function recodes one or more numeric variables into categorical variables based on a specified lower end, upper end, and intermediate breaks. The intervals created include the right endpoint of the interval. For example, breaks = c(2, 3) with lower_end = 1 and upper_end = 5 creates intervals from 1 to <= 2, >2 to <= 3, and >3 to <= 5. If the lower or upper ends are not provided, the function defaults to the minimum and maximum values of the data and issues a warning. This default behavior is prone to errors, however, because a scale may not include its actual lower and upper ends which might in turn affect the recoding process. Hence, it is strongly suggested to manually set the lower and upper bounds of the original continuous scale.

Usage

```
categorize_scale(
  data,
  ...,
  breaks,
  labels,
  lower_end = NULL,
  upper_end = NULL,
  name = NULL,
  overwrite = FALSE
)
```

Arguments

`data`	A tibble or a tdcmm model.
`...`	Variables to recode as factor variables in categories. If no variables are specified, all numeric columns will be recoded.
`breaks`	A vector of numeric values specifying the breaks for categorizing the data between the lower and upper ends. The breaks define the boundaries of the intervals. Setting this parameter is required.
`labels`	A vector of string labels for each interval. The number of labels must match the number of intervals defined by the breaks and lower/upper ends.Setting this parameter is required.
`lower_end`	Optional numeric value specifying the lower end of the scale. If not provided, defaults to the minimum value of the data.
`upper_end`	Optional numeric value specifying the upper end of the scale. If not provided, defaults to the maximum value of the data.
`name`	Optional string specifying the name of the new variable(s). By default, the new variable names are the original variable names suffixed with `_cat`.
`overwrite`	Logical indicating whether to overwrite the original variable(s) with the new categorical variables. If `TRUE`, the original variable(s) are overwritten.

Value

A modified tibble or tdcmm model with the recoded variables.

See Also

Other scaling: `center_scale()`, `dummify_scale()`, `minmax_scale()`, `recode_cat_scale()`, `reverse_scale()`, `setna_scale()`, `z_scale()`

Examples

```
WoJ %>%
dplyr::select(trust_parliament, trust_politicians) %>%
categorize_scale(trust_parliament, trust_politicians,
lower_end = 1, upper_end = 5, breaks = c(2, 3),
labels = c("Low", "Medium", "High"), overwrite = FALSE)
WoJ %>%
dplyr::select(autonomy_selection) %>%
categorize_scale(autonomy_selection, breaks = c(2, 3, 4),
lower_end = 1, upper_end = 5,
labels = c("Low", "Medium", "High", "Very High"),
name = "autonomy_in_categories")
```

`center_scale` *Center numeric, continuous variables*

Description

This function centers the specified numeric columns or all numeric columns if none are specified. A centered scale has a mean of 0.0.

Usage

```
center_scale(data, ..., name = NULL, overwrite = FALSE)
```

Arguments

`data`	A tibble or a tdcmm model.
`...`	Numeric variables to be centered. If none are provided, all numeric columns will be centered.
`name`	Optional name for the new centered variable when a single variable is provided. By default, the name will be the original variable name suffixed with `_centered`.
`overwrite`	Logical. If `TRUE`, it overwrites the original variable(s) with the centered values. If `FALSE` (default), a new variable(s) is created.

Value

A tdcmm model with the centered variable(s).

See Also

Other scaling: `categorize_scale()`, `dummify_scale()`, `minmax_scale()`, `recode_cat_scale()`, `reverse_scale()`, `setna_scale()`, `z_scale()`

Examples

```
WoJ %>% dplyr::select(autonomy_emphasis) %>% center_scale(autonomy_emphasis)
WoJ %>% center_scale(autonomy_emphasis, name = "my_centered_variable")
WoJ %>% center_scale(overwrite = TRUE)
WoJ %>%
  center_scale(autonomy_emphasis) %>%
  tab_frequencies(autonomy_emphasis, autonomy_emphasis_centered)
```

`correlate` *Compute correlation coefficients*

Description

Computes correlation coefficients for all combinations of the specified variables. If no variables are specified, all numeric (integer or double) variables are used.

Usage

```
correlate(data, ..., method = "pearson", partial = NULL, with = NULL)
```

Arguments

`data`	a tibble or a tdcmm model
`...`	Variables to compute correlations for (column names). Leave empty to compute for all numeric variables in data.
`method`	a character string indicating which correlation coefficient is to be computed. One of "pearson" (default), "kendall", or "spearman"
`partial`	Specifies a variable to be used as a control in a partial correlation. By default, this parameter is set to `NULL`, indicating that no control variable is used in the correlation. If used, `with` must be set to `NULL` (default).
`with`	Specifies a focus variable to correlate all other variables with. By default, this parameter is set to `NULL`, indicating that no focus variable is used in the correlation. If used, `partial` must be set to `NULL` (default).

Value

a tdcmm model

Examples

```
WoJ %>% correlate(ethics_1, ethics_2, ethics_3)
WoJ %>% correlate()
WoJ %>% correlate(ethics_1, ethics_2, ethics_3, with = work_experience)
WoJ %>% correlate(autonomy_selection, autonomy_emphasis, partial = work_experience)
WoJ %>% correlate(with = work_experience)
```

`crosstab` *Crosstab variables*

Description

Computes contingency table for one independent (column) variable and one or more dependent (row) variables.

Usage

```
crosstab(
  data,
  col_var,
  ...,
  add_total = FALSE,
  percentages = FALSE,
  chi_square = FALSE
)
```

Arguments

`data`	a tibble or a tdcmm model
`col_var`	Independent (column) variable.
`...`	Dependent (row) variables.
`add_total`	Logical indicating whether a 'Total' column should be computed. Defaults to `FALSE`.
`percentages`	Logical indicating whether to output column-wise percentages instead of absolute values. Defaults to `FALSE`.
`chi_square`	Logical indicating whether a Chi-square test should be computed. Test results will be reported via message(). Defaults to `FALSE`.

Value

a tdcmm model

See Also

Other categorical: `tab_frequencies()`

Examples

```
WoJ %>% crosstab(reach, employment)
WoJ %>% crosstab(reach, employment, add_total = TRUE, percentages = TRUE, chi_square = TRUE)
```

`describe` *Describe numeric variables*

Description

Describe numeric variables by several measures of central tendency and variability. If no variables are specified, all numeric (integer or double) variables are described.

Usage

```
describe(data, ..., na.rm = TRUE)
```

Arguments

`data`	a tibble or a tdcmm model
`...`	Variables to describe (column names). Leave empty to describe all numeric variables in data.
`na.rm`	a logical value indicating whether `NA` values should be stripped before the computation proceeds. Defaults to `TRUE`.

Details

- N: number of valid cases (i.e., all but missing)
- Missing: number of NA cases
- M: mean average
- SD: standard deviation, sd
- Min: minimum value, min
- Q25: 25% quantile, quantile
- Mdn: median average, same as 50% quantile
- Q75: 75% quantile, quantile
- Max: maximum value, max
- Range: difference between Min and Max
- CI_95_LL: $M - Q(0.975) \times \frac{SD}{\sqrt{N}}$ where $Q(0.975)$ denotes Student t's stats::quantile function with a probability of 0.975 and $N - 1$ degrees of freedom
- CI_95_UL: $M + Q(0.975) \times \frac{SD}{\sqrt{N}}$ where $Q(0.975)$ denotes Student t's stats::quantile function with a probability of 0.975 and $N - 1$ degrees of freedom

- Skewness: traditional Fisher-Pearson coefficient of skewness of valid cases as per $\frac{\frac{1}{N}\sum\limits_{i=1}^{N}(x_i - \overline{x})^3}{[\frac{1}{N}\sum\limits_{i=1}^{N}(x_i - \overline{x})^2]^{3/2}}$ where $\overline{x}$ denotes M, following Doane & Seward (2011, p. 6, 1a). See DOI doi:10.1080/10691898.2011.11889611.
- Kurtosis: empirical sample kurtosis (i.e., standardized fourth population moment about the mean) as per $\frac{\sum(x-\overline{x})^4/N}{(\sum(x-\overline{x})^2/N)^2}$, following DeCarlo (1997, p. 292, b2). See DOI doi:10.1037/1082989X.2.3.292.

Value

a tdcmm model

See Also

Other descriptives: `describe_cat()`, `tab_percentiles()`

Examples

```
WoJ %>% describe(autonomy_selection, autonomy_emphasis, work_experience)
fbposts %>% describe(n_pictures)
```

describe_cat *Describe categorical variables*

Description

Describe categorical variables by N, number of unique values, and mode. Note that in case of multiple modes, the first mode by order of values is chosen.

Usage

```
describe_cat(data, ...)
```

Arguments

`data`	a tibble or a tdcmm model
`...`	Variables to describe (column names). Leave empty to describe all categorical variables in data.

Details

If no variables are specified, all categorical (character or factor) variables are described.

- N: number of valid cases (i.e., all but missing)
- Missing: number of NA cases
- Unique: number of unique categories in a given variable, without Missing
- Mode: mode average (if multiple modes exist, first mode by order of values is returned)
- Mode_N: number of cases reflecting the Mode

Value

a tdcmm model

See Also

Other descriptives: `describe()`, `tab_percentiles()`

Examples

```
WoJ %>% describe_cat(reach, employment, temp_contract)
fbposts %>% describe_cat(type)
```

`design_gray` *Gray design*

Description

Gray design

Usage

```
design_gray()
```

Value

a list with `main_color_1`, a vector of 12 `main_colors`, a corresponding `main_contrast_1` (the color of text to write on top of the main color) and a corresponding `main_contrasts`, the `main_size` (for lines), a `comparison_linetype`, `comparison_color`, and `comparison_size` for all lines that act as comparative lines, and a theme()

`design_grey` *Grey design*

Description

Grey design

Usage

```
design_grey()
```

Value

a list with `main_color_1`, a vector of 12 `main_colors`, a corresponding `main_contrast_1` (the color of text to write on top of the main color) and a corresponding `main_contrasts`, the `main_size` (for lines), a `comparison_linetype`, `comparison_color`, and `comparison_size` for all lines that act as comparative lines, and a theme()

`design_lmu` *Colorbrewer-inspired design with focus on LMU (lmu.de) green*

Description

Colorbrewer-inspired design with focus on LMU (lmu.de) green

Usage

```
design_lmu()
```

Value

a list with `main_color_1`, a vector of 12 `main_colors`, a corresponding `main_contrast_1` (the color of text to write on top of the main color) and a corresponding `main_contrasts`, the `main_size` (for lines), a `comparison_linetype`, `comparison_color`, and `comparison_size` for all lines that act as comparative lines, and a theme()

dummify_scale — *Convert categorical variables to dummy variables*

Description

This function transforms specified categorical variables into dummy variables. Each level of the categorical variable is represented by a new dummy variable. Missing values are retained. These new dummy variables are appended to the original data frame. This function does not allow specifying new column names for the dummy variables. Instead, it follows a consistent naming pattern: the new dummy variables are named using the original variable name with the category value appended. For example, if a categorical variable named "autonomy" with levels "low", "medium", "high" is dummified, the new dummy variables will be named "autonomy_low", "autonomy_medium", "autonomy_high".

Usage

```
dummify_scale(data, ..., overwrite = FALSE)
```

Arguments

`data`	A tibble or a tdcmm model.
`...`	Categorical variables to be transformed into dummy variables. Category names will be automatically appended to the newly created dummy variables.
`overwrite`	Logical. If `TRUE`, it overwrites the original variable(s) with the dummy variables. If `FALSE` (default), new variables are created.

Value

A tdcmm model with the dummy variables appended.

See Also

Other scaling: `categorize_scale()`, `center_scale()`, `minmax_scale()`, `recode_cat_scale()`, `reverse_scale()`, `setna_scale()`, `z_scale()`

Examples

```
WoJ %>% dplyr::select(temp_contract) %>% dummify_scale(temp_contract)
WoJ %>% categorize_scale(autonomy_emphasis, breaks = c(2, 3),
labels = c('low', 'medium', 'high')) %>%
dummify_scale(autonomy_emphasis_cat) %>% dplyr::select(starts_with('autonomy_emphasis'))
```

`fbposts` *Facebook posts reliability test*

Description

45 political facebook posts coded by 6 coders for an intercoder reliability test, focused on populist messages.

Usage

```
fbposts
```

Format

A data frame with 270 rows and 7 variables

post_id Numeric id of the coded Facebook post

coder_id Numeric id of the coder

type Type of Facebook post, one of "link", "photo", "status", or "video

n_pictures Amount of pictures attached to the post, ranges from 0 to 6

pop_elite Populism indicator: Does the Facebook post attack elites?, 0 = "no attacks on elites", 1 = "attacks political actors", 2 = "attacks public administration actors", 3 = "attacks economical actors", 4 = "attacks media actors/journalists", 9 = "attacks other elites"

pop_people Populism indicator: Does the Facebook refer to 'the people'?, 0 = "does not refer to 'the people'", 1 = "refers to 'the people'"

pop_othering Populism indicator: Does the Facebook attack 'others'?, 0 = "no attacks on 'others'", 1 = "attacks other cultures", 2 = "attacks other political stances", 3 = "attacks other 'others'"

`get_reliability` *Get reliability estimates of index variables*

Description

Get reliability estimates of index variables created with `add_index`.

Usage

```
get_reliability(
  data,
  ...,
  type = "alpha",
  interval.type = NULL,
  bootstrap.samples = NULL,
  conf.level = NULL,
  progress = FALSE
)
```

Arguments

data	a tibble or a tdcmm model
...	Index variables created with `add_index`. Leave empty to get reliability estimates for all index variables.
type	Type of reliability estimate. See `ci.reliability`
interval.type	Type of reliability estimate confidence interval. See `ci.reliability`
bootstrap.samples	Number of bootstrap samples for CI calculation. See `ci.reliability`
conf.level	Confidence level for estimate CI. See `ci.reliability`
progress	Show progress for reliability estimate computation. Useful if using computationally intense computations (e. g., many bootstrapping samples) and many index variables.

Value

a tdcmm model

See Also

`add_index()` to create index variables

Examples

```
WoJ %>%
  add_index(ethical_flexibility, ethics_1, ethics_2, ethics_3, ethics_4) %>%
  get_reliability()
```

`incvlcomments` *Incivil Comments Data*

Description

A dataset of a preregistered factorial survey experiment with a nationally representative sample of 964 German online users. Participants were presented with manipulated user comments that included statements associated with incivil discourse (such as profanity and attacks on arguments) and intolerant discourse (such as offensive stereotyping and violent threats). Participants rated the comments, e.g. offensiveness, harm to society, and their intention to delete the comment containing the statement.

Usage

```
incvlcomments
```

Format

A data frame of 3856 observations nested in 964 participants and 22 variables:

participant_num Numeric id of the participant

age Age of the participant

male Gender of the participant, either 'male' or 'not male'

high_education Level of formal education of the participant, either 'high formal education' or 'low formal education'

comment_num Numeric id of the comment that the participant was exposed to

issue The subject of the comment that the participant was exposed to, either 'Gender', 'Abortion', 'Climate', or 'Migration'

profanity Whether the comment contained profanities as an indicator of incivility

attacks_argument Whether the comment contained attacks towards arguments as an indicator of incivility

offensive_stereotyping Whether the comment contained offensive stereotypes as an indicator of intolerant discourse

violent_threats Whether the comment contained violent threats as an indicator of intolerant discourse

offensiveness Rate statement whether the comment is being perceived as offensive & hostile (Scale from 1 to 7)

adequacy Rate statement whether the comment is being perceived as necessary & accurate (Scale from 1 to 7)

harm_to_society Rate statement whether the comment is being perceived as harmful to society (Scale from 1 to 7)

deletion_intention Whether the participant wants to delete the comment

similarity_poster How similar the participant feels to the person who created the post (Scale from 1 to 7)

similarity_group How similar the participant feels to the group of people criticized in the post (Scale from 1 to 7)

attitude_gender Rate agreement with statements on gender policies (Scale from 1 to 7)

attitude_abortion Rate agreement with statements on abortion (Scale from 1 to 7)

attitude_migration Rate agreement with statements on migration (Scale from 1 to 7)

attitude_climate Rate agreement with statements on climate change (Scale from 1 to 7)

left_right_placement Placement on a political spectrum from left to right (Scale from 1 to 9)

freedom_of_speech Rate agreement with statements about the freedom of speech and expression (Scale from 1 to 7)

Details

The dataset was created from the OSF project: Differential perceptions of and reactions to incivil and intolerant user comments, corresponding to the paper: Kümpel, A. S., Unkel, J (2023). Differential perceptions of and reactions to incivil and intolerant user comments, Journal of Computer-Mediated Communication, Volume 28, Issue 4, https://doi.org/10.1093/jcmc/zmad018

Source

https://osf.io/w92vj

`minmax_scale` *Rescale numeric continuous variables to new minimum/maximum boundaries*

Description

Given a specified minimum and maximum, this function translates each value into a new value within this specified range. The transformation maintains the relative distances between values, resulting in changes to the mean and standard deviations. However, if both the original scale and the transformed scale are z-standardized, they will be equal again, indicating that the relative positions and distributions of the values remain consistent.

Usage

```
minmax_scale(
  data,
  ...,
  change_to_min = 0,
  change_to_max = 1,
  name = NULL,
  overwrite = FALSE
)
```

Arguments

`data`	A tibble or a tdcmm model.
`...`	Numeric variables to be min-max scaled. If none are provided, all numeric columns will be scaled.
`change_to_min`	The desired minimum value after scaling.
`change_to_max`	The desired maximum value after scaling.
`name`	Optional name for the new scaled variable when a single variable is provided. By default, the name will be the original variable name suffixed with the range. For example, "variable" becomes "variable_3to5". Negative values are prefixed with "neg" to avoid invalid columns names (e.g., -3 to 3 becomes "variable_neg3to5").
`overwrite`	Logical. If `TRUE`, it overwrites the original variable(s) with the scaled values. If `FALSE` (default), a new variable(s) is created.

Value

A tdcmm model with the min-max scaled variable(s).

See Also

Other scaling: `categorize_scale()`, `center_scale()`, `dummify_scale()`, `recode_cat_scale()`, `reverse_scale()`, `setna_scale()`, `z_scale()`

Examples

```
WoJ %>% minmax_scale(autonomy_emphasis, change_to_min = 0,
change_to_max = 1)
WoJ %>% minmax_scale(autonomy_emphasis, name = "my_scaled_variable",
change_to_min = 0, change_to_max = 1)
WoJ %>%
  minmax_scale(autonomy_emphasis, change_to_min = 0, change_to_max = 1) %>%
  tab_frequencies(autonomy_emphasis, autonomy_emphasis_0to1)
```

`recode_cat_scale` *Recode one or more categorical variables into new categories*

Description

This function transforms one or more categorical variables into new categories based on specified mapping. For unmatched cases not specified in the mapping, a default value can be assigned. Missing values are retained.

Usage

```
recode_cat_scale(
  data,
  ...,
  assign = NULL,
  other = NA,
  overwrite = FALSE,
  name = NULL
)
```

Arguments

`data`	A tibble or a tdcmm model.
`...`	Variables to recode.
`assign`	A named vector where names are the old values and values are the new values to be assigned.
`other`	The value for unmatched cases. By default, it is `NA`. This parameter is used to assign a value to cases that do not match any of the keys in the `assign` vector.
`overwrite`	Logical. If `TRUE`, it overwrites the original variable(s). You cannot specify both 'name' and 'overwrite' parameters simultaneously.
`name`	The name of the new variable(s). If not specified, this is the same name as the provided variable(s) but suffixed with `_rec`.

Value

A tdcmm model or a tibble.

See Also

Other scaling: `categorize_scale()`, `center_scale()`, `dummify_scale()`, `minmax_scale()`, `reverse_scale()`, `setna_scale()`, `z_scale()`

Examples

```
WoJ %>%
recode_cat_scale(country,
assign = c("Germany" = 1, "Switzerland" = 2), overwrite = TRUE)
WoJ %>%
recode_cat_scale(country,
assign = c("Germany" = "german", "Switzerland" = "swiss"), other = "other",
overwrite = TRUE)
WoJ %>%
recode_cat_scale(ethics_1, ethics_2,
assign = c(`1` = 5, `2` = 4, `3` = 3, `4` = 2, `5` = 1), other = 6, overwrite = TRUE)
WoJ %>%
recode_cat_scale(ethics_1, ethics_2,
assign = c(`1` = "very low", `2` = "low", `3` = "medium", `4` = "high", `5` = "very high"),
overwrite = TRUE)
WoJ %>%
dplyr::select(temp_contract) %>% recode_cat_scale(temp_contract,
assign = c(`Permanent` = "P", `Temporary` = "T"), other = "O")
```

regress *Compute linear regression*

Description

Computes linear regression for all independent variables on the specified dependent variable. Linear modeling of multiple independent variables uses stepwise regression modeling. If specified, preconditions for (multi-)collinearity and for homoscedasticity are checked.

Usage

```
regress(
  data,
  dependent_var,
  ...,
  check_independenterrors = FALSE,
  check_multicollinearity = FALSE,
  check_homoscedasticity = FALSE
)
```

Arguments

`data`	a tibble or a tdcmm model
`dependent_var`	The dependent variable on which the linear model is fitted. Specify as column name.
`...`	Independent variables to take into account as (one or many) predictors for the dependent variable. Specify as column names. At least one has to be specified.
`check_independenterrors`	if set, the independence of errors among any two cases is being checked using a Durbin-Watson test
`check_multicollinearity`	if set, multicollinearity among all specified independent variables is being checked using the variance inflation factor (VIF) and the tolerance (1/VIF); this check can only be performed if at least two independent variables are provided, and all provided variables need to be numeric
`check_homoscedasticity`	if set, homoscedasticity is being checked using a Breusch-Pagan test

Value

a tdcmm model

Examples

```
WoJ %>% regress(autonomy_selection, ethics_1)
WoJ %>% regress(autonomy_selection, work_experience, trust_government)
```

`reverse_scale` *Reverse numeric, logical, or date/time continuous variables*

Description

Reverses a continuous scale into a new variable. A 5-1 scale thus turns into a 1-5 scale. Missing values are retained. For a given continuous variable the lower and upper end of the scale should be provided. If they are not provided, the function assumes the scale's minimum and maximum value to represent these lower/upper ends (and issues a warning about this fact). This default behavior is prone to errors, however, because a scale may not include its actual lower and upper ends which might in turn affect correct reversing. Hence, it is strongly suggested to manually set the lower and upper bounds of the original continuous scale.

Usage

```
reverse_scale(
  data,
  ...,
  lower_end = NULL,
  upper_end = NULL,
  name = NULL,
  overwrite = FALSE
)
```

Arguments

data	A tibble or a tdcmm model.
...	Numeric variables to be reverse scaled. If none are provided, all numeric columns will be scaled.
lower_end	Lower end of provided continuous scale (default is to use minimum value of current values, which might not be the actual lower end of the scale).
upper_end	Upper end of provided continuous scale (default is to use maximum value of current values, which might not be the actual upper end of the scale).
name	Optional name for the new reversed variable when a single variable is provided. By default, the name will be the original variable name suffixed with `_rev`.
overwrite	Logical. If `TRUE`, it overwrites the original variable(s) with the reversed values. If `FALSE` (default), a new variable(s) is created.

Value

A tdcmm model with the reversed variable(s).

See Also

Other scaling: `categorize_scale()`, `center_scale()`, `dummify_scale()`, `minmax_scale()`, `recode_cat_scale()`, `setna_scale()`, `z_scale()`

Examples

```
WoJ %>% reverse_scale(autonomy_emphasis, lower_end = 0, upper_end = 1)
WoJ %>% reverse_scale(autonomy_emphasis, name = "my_reversed_variable",
lower_end = 0, upper_end = 1)
WoJ %>% reverse_scale(overwrite = TRUE)
WoJ %>%
  reverse_scale(autonomy_emphasis, lower_end = 0, upper_end = 1) %>%
  tab_frequencies(autonomy_emphasis, autonomy_emphasis_rev)
```

---

setna_scale — *Set specified values to NA in selected variables or entire data frame*

Description

This function allows users to set specific values to `NA` in chosen variables within a data frame. It can handle numeric, character, and factor variables.

Usage

```
setna_scale(data, ..., value, name = NULL, overwrite = FALSE)
```

Arguments

`data`	A tibble or a tdcmm model.
`...`	One or more variables where specified values will be set to NA. If no variables are provided, the function is applied to the entire data frame.
`value`	A value (or vector of values) that needs to be set to NA.
`name`	The name of the new variable(s). By default, this is the same name as the provided variable(s) but suffixed with `_na`.
`overwrite`	Logical. If `TRUE`, it overwrites the original variable(s). You cannot specify both 'name' and 'overwrite' parameters simultaneously.

Value

A tdcmm model or a tibble.

See Also

Other scaling: `categorize_scale()`, `center_scale()`, `dummify_scale()`, `minmax_scale()`, `recode_cat_scale()`, `reverse_scale()`, `z_scale()`

Examples

```
WoJ %>%
dplyr::select(autonomy_emphasis) %>%
setna_scale(autonomy_emphasis, value = 5)
WoJ %>%
dplyr::select(autonomy_emphasis) %>%
setna_scale(autonomy_emphasis, value = 5, name = "new_na_autonomy")
WoJ %>%
setna_scale(value = c(2, 3, 4), overwrite = TRUE)
WoJ %>%
dplyr::select(country) %>% setna_scale(country, value = "Germany")
WoJ %>%
dplyr::select(country) %>% setna_scale(country, value = c("Germany", "Switzerland"))
```

`snscomments` *SNS Comments data*

Description

A dataset of 630 German participants in an online experiment. The experiment investigated the effects of user comments on social network sites (SNS) on individuals' perceptions of journalistic quality. The researchers varied the subject of the article (factor 1: 'Copyright directive' or 'Social housing'), the order of comment presentation (factor 2: before or after the article) and the valence of the comments (factor 3: positive or negative).

Usage

```
snscomments
```

Format

A data frame of 630 observations and 15 variables:

age Age of the participant

gender Gender of the participant, either 'female' or 'not female'

education Level of formal education of the participant, either 'low formal education' or 'high formal education'

need_cognition Index measuring the psychological trait of a person to enjoy thinking, calculated from several survey items

prior_knowledge Index measuring a person's prior knowledge of the presented subject of the article, calculated from several survey items

group Numeric id of the group that the participant was in during the experiment

issue Subject of the article that the participant was given to read, either 'Copyright directive' or 'Social housing'

order Order of the comments that the participant was exposed to, either 'Comments after', 'Comments before', or 'Control group'

valence Valence of the comments that the participant was exposed to, either 'Negative', 'Positive', or 'Control group'

control_group Indicates whether the participant was in the 'Control group' or 'Experimental group'

medium_evaluation Index measuring participant's evaluation of the medium's quality, calculated from several survey items

article_evaluation Index measuring participant's evaluation of the article's quality, calculated from several survey items

comments_quality Participant's perception of the quality of the comments

comments_valence Participant's perception of the valence of the comments

article_elaboration Participant's measure of how much attention they put in reading the article

Details

This dataset was created from the OSF project: `https://osf.io/r867v/`, corresponding to the paper: Kümpel, A. S., & Unkel, J. (2020). Negativity wins at last: How presentation order and valence of user comments affect perceptions of journalistic quality. Journal of Media Psychology: Theories, Methods, and Applications, 32(2), 89–99. doi:10.1027/18641105/a000261

Source

`https://osf.io/r867v/`

`tab_frequencies` *Tabulate frequencies*

Description

Tabulates frequencies for one or more categorical variable, including relative, and cumulative frequencies.

Usage

```
tab_frequencies(data, ...)
```

Arguments

`data`	a tibble or a tdcmm model
`...`	Variables to tabulate

Value

a tdcmm model

See Also

Other categorical: `crosstab()`

Examples

```
WoJ %>% tab_frequencies(employment)
WoJ %>% tab_frequencies(employment, country)
```

`tab_percentiles` *Tabulate percentiles for numeric variables*

Description

This function tabulates specified percentiles for given numeric variables. If no variables are provided, the function will attempt to describe all numeric (either integer or double) variables found within the input. The percentiles are calculated based on the levels parameter, which defaults to every 10% from 10% to 90%. NA values are always removed because the concept of a percentile is based on ranking. As NA is not a value, it cannot be ordered in relation to actual numbers.

Usage

```
tab_percentiles(
  data,
  ...,
  levels = c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1)
)
```

Arguments

`data`	a tibble or a tdcmm model that contains the numeric data to be tabulated.
`...`	Variables within the data for which to tabulate the percentiles. If no variables are provided, all numeric variables are used.
`levels`	a numeric vector specifying the percentiles to compute. Defaults to c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0).

Value

a tdcmm model

See Also

Other descriptives: `describe()`, `describe_cat()`

Examples

```
WoJ %>% tab_percentiles(work_experience)
WoJ %>% tab_percentiles(work_experience, autonomy_emphasis)
```

`test_icr` *Perform an intercoder reliability test*

Description

Performs an intercoder reliability test by computing various intercoder reliability estimates for the included variables

Usage

```
test_icr(
  data,
  unit_var,
  coder_var,
  ...,
  levels = NULL,
  na.omit = FALSE,
  agreement = TRUE,
  holsti = TRUE,
  kripp_alpha = TRUE,
  cohens_kappa = FALSE,
  fleiss_kappa = FALSE,
  brennan_prediger = FALSE,
  lotus = FALSE,
  s_lotus = FALSE
)
```

Arguments

`data`	a tibble or a tdcmm model
`unit_var`	Variable with unit identifiers
`coder_var`	Variable with coder identifiers
`...`	Variables to compute intercoder reliability estimates for. Leave empty to compute for all variables (excluding `unit_var` and 'coder_var") in data.
`levels`	Optional named vector with levels of test variables
`na.omit`	Logical indicating whether `NA` values should be stripped before computation. Defaults to `FALSE`.
`agreement`	Logical indicating whether simple percent agreement should be computed. Defaults to `TRUE`.
`holsti`	Logical indicating whether Holsti's reliability estimate (mean pairwise agreement) should be computed. Defaults to `TRUE`.
`kripp_alpha`	Logical indicating whether Krippendorff's Alpha should be computed. Defaults to `TRUE`.
`cohens_kappa`	Logical indicating whether Cohen's Kappa should be computed. Defaults to `FALSE`.
`fleiss_kappa`	Logical indicating whether Fleiss' Kappa should be computed. Defaults to `FALSE`.
`brennan_prediger`	Logical indicating whether Brennan & Prediger's Kappa should be computed (extension to 3+ coders as proposed by von Eye (2006)). Defaults to `FALSE`.
`lotus`	Logical indicating whether Fretwurst's Lotus should be computed. Defaults to `FALSE`
`s_lotus`	Logical indicating whether Fretwurst's standardized Lotus (S-Lotus) should be computed. Defaults to `FALSE`.

Value

a tdcmm model

References

Brennan, R. L., & Prediger, D. J. (1981). Coefficient Kappa: Some uses, misuses, and alternatives. Educational and Psychological Measurement, 41(3), 687-699. https://doi.org/10.1177/001316448104100307

Cohen, J. (1960). A coefficient of agreement for nominal scales. Educational and Psychological Measurement, 20(1), 37-46. https://doi.org/10.1177/001316446002000104

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. Psychological Bulletin, 76(5), 378-382. https://doi.org/10.1037/h0031619

Fretwurst, B. (2015). Reliabilität und Validität von Inhaltsanalysen. Mit Erläuterungen zur Berechnung des Reliabilitätskoeffizienten „Lotus“ mit SPSS. In W. Wirth, K. Sommer, M. Wettstein, & J. Matthes (Ed.), Qualitätskriterien in der Inhaltsanalyse (S. 176–203). Herbert von Halem.

Krippendorff, K. (2011). Computing Krippendorff's Alpha-Reliability. Retrieved from http://repository.upenn.edu/asc_paper

von Eye, A. (2006). An Alternative to Cohen's Kappa. European Psychologist, 11(1), 12-24. https://doi.org/10.1027/1016-9040.11.1.12

Examples

```
fbposts %>% test_icr(post_id, coder_id, pop_elite, pop_othering)
fbposts %>% test_icr(post_id, coder_id, levels = c(n_pictures = "ordinal"), fleiss_kappa = TRUE)
```

`to_correlation_matrix` *Create correlation matrix*

Description

Turns the tibble exported from `correlate` into a correlation matrix.

Usage

```
to_correlation_matrix(data, verbose = FALSE)
```

Arguments

`data`	a tdcmm model returned from `correlate`
`verbose`	A logical, defaulted to `FALSE`. Only applicable when correlating two variables. If set to `TRUE`, the function outputs information regarding the sample size.

Value

a tdcmm model

Examples

```
WoJ %>% correlate() %>% to_correlation_matrix()
```

`t_test` *Compute t-tests*

Description

Computes t-tests for one group variable and specified test variables. If no variables are specified, all numeric (integer or double) variables are used. A Levene's test will automatically determine whether the pooled variance is used to estimate the variance. Otherwise the Welch (or Satterthwaite) approximation to the degrees of freedom is used.

Usage

```
t_test(
  data,
  group_var,
  ...,
  var.equal = TRUE,
  paired = FALSE,
  pooled_sd = TRUE,
  levels = NULL,
  case_var = NULL,
  mu = NULL
)
```

Arguments

`data`	a tibble or a tdcmm model
`group_var`	group variable (column name) to specify where to split two samples (two-sample t-test) or which variable to compare a one-sample t-test on
`...`	test variables (column names). Leave empty to compute t-tests for all numeric variables in data. Also leave empty for one-sample t-tests.
`var.equal`	this parameter is deprecated (previously: a logical variable indicating whether to treat the two variances as being equal. If `TRUE` then the pooled variance is used to estimate the variance otherwise the Welch (or Satterthwaite) approximation to the degrees of freedom is used. Defaults to `TRUE`).
`paired`	a logical indicating whether you want a paired t-test. Defaults to `FALSE`.
`pooled_sd`	a logical indicating whether to use the pooled standard deviation in the calculation of Cohen's d. Defaults to `TRUE`.
`levels`	optional: a vector of length two specifying the two levels of the group variable.
`case_var`	optional: case-identifying variable (column name). If you set `paired = TRUE`, specifying a case variable will ensure that data are properly sorted for a dependent t-test.
`mu`	optional: a number indicating the *true* value of the mean in the general population (μ). If set, a one-sample t-test (i.e., a location test) is being calculated. Leave to `NULL` to calculate two-sample t-test(s).

Value

a tdcmm model

Examples

```
WoJ %>% t_test(temp_contract, autonomy_selection, autonomy_emphasis)
WoJ %>% t_test(temp_contract)
WoJ %>% t_test(employment, autonomy_selection, autonomy_emphasis,
  levels = c("Full-time", "Freelancer"))
WoJ %>% t_test(autonomy_selection, mu = 3.62)
```

`unianova` *Compute one-way ANOVAs*

Description

Computes one-way ANOVAs for one group variable and specified test variables. If no variables are specified, all numeric (integer or double) variables are used. A Levene's test will automatically determine whether a classic ANOVA is used. Otherwise Welch's ANOVA with a (Satterthwaite's) approximation to the degrees of freedom is used.

Usage

```
unianova(data, group_var, ..., descriptives = FALSE, post_hoc = FALSE)
```

Arguments

`data`	a tibble or a tdcmm model
`group_var`	group variable (column name)
`...`	test variables (column names). Leave empty to compute ANOVAs for all numeric variables in data.
`descriptives`	a logical indicating whether descriptive statistics (mean & standard deviation) for all group levels should be added to the returned tibble. Defaults to `FALSE`.
`post_hoc`	a logical value indicating whether post-hoc tests should be performed. Tukey's HSD is employed when the assumption of equal variances is met, whereas the Games-Howell test is automatically applied when this assumption is violated. The results of the post-hoc test will be added to a list column in the resulting tibble.

Value

a tdcmm model

Examples

```
WoJ %>% unianova(employment, autonomy_selection, autonomy_emphasis)
WoJ %>% unianova(employment, descriptives = TRUE, post_hoc = TRUE)
## Not run:
WoJ %>% unianova(employment)

## End(Not run)
```

`visualize.tdcmm_ctgrcl` *Visualize tidycomm output*

Description

Returns ggplot visualization appropriate to respective `tdcmm` model (see list below). Returns `NULL` (and a warning) if no visualization has been implemented for the particular model.

Usage

```
## S3 method for class 'tdcmm_ctgrcl'
visualize(x, ..., .design = design_lmu())

## S3 method for class 'tdcmm_crrltn'
visualize(x, which = "jitter", ..., .design = design_lmu())

## S3 method for class 'tdcmm_dscrb'
visualize(x, ..., .design = design_lmu())

## S3 method for class 'tdcmm_rgrssn'
visualize(x, which = "jitter", ..., .design = design_lmu())

## S3 method for class 'tdcmm_prcntl'
visualize(x, ..., .design = design_lmu())

visualize(x, ..., .design = design_lmu())

## S3 method for class 'tdcmm_ttst'
visualize(x, ..., .design = design_lmu())

## S3 method for class 'tdcmm_nnv'
visualize(x, ..., .design = design_lmu())
```

Arguments

`x`	`tdcmm` output
`...`	other arguments

`.design`	a list to style the visualization; by default and good practice use one of the ready-made design functions' returns (e.g., `design_lmu()`, `design_grey()`); you could, however, also provide your own list here which has to be a list with 9 keys: `main_color_1`, a vector of 12 `main_colors`, a corresponding `main_contrast_1` (the color of text to write on top of the main color) and a corresponding `main_contrasts`, the `main_size` (for lines), a `comparison_linetype`, `comparison_color`, and `comparison_size` for all lines that act as comparative lines, and a `ggplot2::theme`
`which`	string to specify type of regression visualization. One of "jitter" (default), "alpha", "correlogram", "residualsfitted" (or "resfit"), "pp", "qq", "scalelocation" (or "scaloc"), "residualsleverage" (or "reslev"). See below for details.

Details

- `describe()`: horizontal box plot depicting a box from Q25 to Q75, a thick line for Mdn, and two whiskers to Min/Max respectively; no additional arguments
- `describe_cat()`: horizontal bar plot depicting number of occurrences; no additional arguments
- `tab_frequencies()`: either a histogram (if 1 variable is given) or multiple histograms wrapped, 5+ variables issue a warning about readability; no additional arguments
- `tab_percentiles()`: quantile plot
- `crosstab()`: horizontal stacked bar plot, either absolute or relative (depending on the `percentages` argument in `crosstab()`)
- `t_test()`: plot with points and appended 95% confidence intervals; no additional arguments
- `unianova()`: plot with points and appended 95% confidence intervals; no additional arguments
- `correlate()`: plot as scatter; for more than 2 variables, a correlogram is plotted (just like for `to_correlation_matrix()`); use the `which` parameter to select how points are visualized:
  - "jitter" adds a bit of random noise to each point to better reflect categorical values
  - "alpha" depicts points slightly transparent so that multiple points in the same position are more easily visible
- `correlate()`: for partial correlation, a scatter plot with some jitter is plotted using the residuals between the control variable and (a) the dependent as well as (b) the independent variable; no additional arguments
- `to_correlation_matrix()`: plot as correlogram building on `GGally::ggpairs()` with jittered scatter plots in lower half, histograms as diagonals, and correlation coefficients with 95% confidence intervals in upper half
- `regress()`: plot regression results as scatter (without jitter) and an additional depicted model line with including its 95% confidence intervals; alternatively, visual check inspection helpers can be plotted through the `which` parameter which can be set to yield one of the following:
  - "jitter" (default): plots a scatter plot with jitter per independent variable and adds a linear regression line with 95% confidence intervals to it; keep in mind that if you have, say, three independent variables, this visualization shows you three plots with one linear regression for each, so that the three models (i.e., the three colored lines) reflect only the particular combination of one independent and the dependent variable

- "alpha" (default): almost like `jitter` but instead of jitter it plots scatter plots with some transparency so that multiple data points in the same position appear as darker
- "correlogram": like `to_correlation_matrix()`, a correlogram between independent variables are produced to help determine independent errors and multicollinearity
- "residualsfitted" or "resfit": a residuals-versus-fitted plot is useful to determine distributions; for a normal distribution the colored line should ideally fit on the dashed line
- "pp": a (normal) probability-probability plot helps checking for multicollinearity whereby the data (here mostly the center data from within the IQR) should ideally align with the dashed line
- "qq": a (normal) quantile-quantile plot helps checking for multicollinearity but focuses more on outliers; the data should align with the dashed line
- "scalelocation" or "scaloc": a scale-location (sometimes also called a spread-location) plot checks whether residuals are spread equally to help check for homoscedasticity; ideally, the colored line is horizontal and the data spreads more or less randomly
- "residualsleverage" or "reslev": a residuals-versus-leverage plot allows to check for influential outliers affecting the final model more than the rest of the data; ideally, no data is far off compared to the bulk of the the data and thus shows high Cook's distance to the rest; the colored line helps to identify the bulk of the data and the five most-distant outliers are labelled with their case number (i.e., the row number in the dataset); note that 5 is arbitrary here, meaning that they might not be too far off or there might be more than 5 noteworthy outliers in this model; interpret with care

Note that the returned ggplot object can be modified easily by appending or overwriting individual geom's or scale's. See the examples below and the documentation of ggplot.

Value

A ggplot object

Examples

```
## Not run:
WoJ %>%
  describe() %>%
  visualize()

fbposts %>%
  describe_cat() %>%
  visualize()

WoJ %>%
  tab_frequencies(trust_parliament) %>%
  visualize()
fbposts %>%
  tab_frequencies(pop_elite, pop_people, pop_othering) %>%
  visualize()

WoJ %>%
  crosstab(reach, employment) %>%
  visualize()
```

```
fbposts %>%
  crosstab(coder_id, type, percentages = TRUE) %>%
  visualize()

WoJ %>%
  t_test(temp_contract, autonomy_selection, autonomy_emphasis) %>%
  visualize()

WoJ %>%
  unianova(country, autonomy_selection, autonomy_emphasis) %>%
  visualize()

fbposts %>%
  correlate(pop_elite, pop_people) %>%
  visualize()

fbposts %>%
  correlate(pop_elite, pop_people, with = pop_othering) %>%
  visualize()

fbposts %>%
  correlate(pop_elite, pop_people) %>%
  visualize("alpha")

WoJ %>%
  correlate(autonomy_selection, ethics_1, partial = work_experience) %>%
  visualize()

WoJ %>%
  correlate(ethics_1, ethics_2, ethics_3, ethics_4) %>%
  to_correlation_matrix() %>%
  visualize()

r <- WoJ %>% regress(autonomy_selection, temp_contract, work_experience, ethics_2)
r %>% visualize() # same as r %>% visualize("jitter")
r %>% visualize("alpha")
r %>% visualize("correlogram")
r %>% visualize("resfit")
r %>% visualize("pp")
r %>% visualize("qq")
r %>% visualize("scaloc")
r %>% visualize("reslev")

# To overwrite a certain scale or geom, just append as you would with ggplot2
fbposts %>%
  describe_cat() %>%
  visualize() +
    ggplot2::scale_fill_grey()

## End(Not run)
```

WoJ *Worlds of Journalism sample data*

Description

A subset of data from the Worlds of Journalism 2012-16 study containing survey data of 1,200 journalists from five European countries.

Usage

```
WoJ
```

Format

A data frame with 1200 rows and 15 variables:

country Country of residence

reach Reach of medium

employment Current employment situation

temp_contract Type of contract (if current employment situation is either full-time or part-time

autonomy_selection Autonomy in news story selection, scale from 1 (*no freedom at all*) to 5 (*complete freedom*)

autonomy_emphasis Autonomy in news story emphasis, scale from 1 (*no freedom at all*) to 5 (*complete freedom*)

ethics_1 Agreement with statement "Journalists should always adhere to codes of professional ethics, regardless of situation and context", scale from 1 (*strongly disagree*) to 5 (*strongly agree*) (*reverse-coded!*)

ethics_2 Agreement with statement "What is ethical in journalism depends on the specific situation.", scale from 1 (*strongly disagree*) to 5 (*strongly agree*)

ethics_3 Agreement with statement "What is ethical in journalism is a matter of personal judgment.", scale from 1 (*strongly disagree*) to 5 (*strongly agree*)

ethics_4 Agreement with statement "It is acceptable to set aside moral standards if extraordinary circumstances require it.", scale from 1 (*strongly disagree*) to 5 (*strongly agree*)

work_experience Work experience as a journalist in years

trust_parliament Trust placed in parliament, scale from 1 (*no trust at all*) to 5 (*complete trust*)

trust_government Trust placed in government, scale from 1 (*no trust at all*) to 5 (*complete trust*)

trust_parties Trust placed in parties, scale from 1 (*no trust at all*) to 5 (*complete trust*)

trust_politicians Trust placed in politicians in general, scale from 1 (*no trust at all*) to 5 (*complete trust*)

Source

‘https://worldsofjournalism.org/data/data-and-key-tables-2012-2016’

`z_scale` *Z-standardize numeric, continuous variables*

Description

This function z-standardizes the specified numeric columns or all numeric columns if none are specified. A z-standardized scale centers at a mean of 0.0 and has a standard deviation of 1.0, making it comparable to other z-standardized distributions.

Usage

```
z_scale(data, ..., name = NULL, overwrite = FALSE)
```

Arguments

`data`	A tibble or a tdcmm model.
`...`	Numeric variables to be z-standardized. If none are provided, all numeric columns will be z-standardized.
`name`	Optional name for the new z-standardized variable when a single variable is provided. By default, the name will be the original variable name suffixed with `_z`.
`overwrite`	Logical. If `TRUE`, it overwrites the original variable(s) with the z-standardized values. If `FALSE` (default), a new variable(s) is created.

Value

A tdcmm model with the z-standardized variable(s).

See Also

Other scaling: `categorize_scale()`, `center_scale()`, `dummify_scale()`, `minmax_scale()`, `recode_cat_scale()`, `reverse_scale()`, `setna_scale()`

Examples

```
WoJ %>% z_scale(autonomy_emphasis)
WoJ %>% z_scale(autonomy_emphasis, name = "my_zstdized_variable")
WoJ %>%
  z_scale(autonomy_emphasis) %>%
  tab_frequencies(autonomy_emphasis, autonomy_emphasis_z)
```

Index