Package 'collinear'

November 8, 2024

Title Automated Multicollinearity Management

Version 2.0.0

URL https://blasbenito.github.io/collinear/

BugReports https://github.com/blasbenito/collinear/issues

Description Effortless multicollinearity management in data frames with both numeric and categorical variables for statistical and machine learning applications. The package simplifies multicollinearity analysis by combining four robust methods: 1) target encoding for categorical variables (Micci-Barreca, D. 2001 <doi:10.1145/507533.507538>); 2) automated feature prioritization to prevent key variable loss during filtering; 3) pairwise correlation for all variable combinations (numeric-numeric, numeric-categorical, categorical-categorical); and 4) fast computation of variance inflation factors.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports progressr, future.apply, mgcv, rpart, ranger

Suggests future, testthat (>= 3.0.0), spelling

Config/testthat/edition 3

Depends R (>= 4.0)

LazyData true

Language en-US

NeedsCompilation no

Author Blas M. Benito [aut, cre, cph]
(<https://orcid.org/0000-0001-5105-7232>)

Maintainer Blas M. Benito <blasbenito@gmail.com>

Repository CRAN

Date/Publication 2024-11-08 13:50:02 UTC

Contents

`add_white_noise` *Add White Noise to Encoded Predictor*

Description

Internal function to add white noise to a encoded predictor to reduce the risk of overfitting when used in a model along with the response.

Usage

```
add_white_noise(
  df = NULL,
  response = NULL,
  predictor = NULL,
  white_noise = 0,
  seed = 1
)
```

Arguments

`df`	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
`response`	(optional, character string) Name of a numeric response variable in `df`. Default: NULL.
`predictor`	(required, string) Name of a target-encoded predictor. Default: NULL
`white_noise`	(optional; numeric vector) Argument of the methods "mean", "rank", and "loo". Maximum white noise to add, expressed as a fraction of the range of the response variable. Range from 0 to 1. Default: `0`.
`seed`	(optional; integer vector) Random seed to facilitate reproducibility when `white_noise` is not 0. If NULL, the function selects one at random, and the selected seed does not appear in the encoded variable names. Default: 0

Value

data frame

See Also

Other target_encoding_tools: `encoded_predictor_name()`

case_weights *Case Weights for Unbalanced Binomial or Categorical Responses*

Description

Case Weights for Unbalanced Binomial or Categorical Responses

Usage

```
case_weights(x = NULL)
```

Arguments

x	(required, integer, character, or factor vector) binomial, categorical, or factor response variable. Default: NULL

Value

numeric vector: case weights

See Also

Other modelling_tools: model_formula(), performance_score_auc(), performance_score_r2(), performance_score_v()

Examples

```
case_weights(
  x = c(0, 0, 0, 1, 1)
  )

case_weights(
  x = c("a", "a", "b", "b", "c")
  )
```

collinear *Automated multicollinearity management*

Description

Automates multicollinearity management in data frames with numeric and non-numeric predictors by combining four methods:

- **Target Encoding**: When a numeric response is provided and encoding_method is not NULL, it transforms categorical predictors (classes "character" and "factor") to numeric using the response values as reference. See target_encoding_lab() for further details.

- **Preference Order**: When a response of any type is provided via `response`, the association between the response and each predictor is computed with an appropriate function (see `preference_order()` and `f_auto()`), and all predictors are ranked from higher to lower association. This rank is used to preserve important predictors during the multicollinearity filtering.
- **Pairwise Correlation Filtering**: Automated multicollinearity filtering via pairwise correlation. Correlations between numeric and categoricals predictors are computed by target-encoding the categorical against the predictor, and correlations between categoricals are computed via Cramer's V. See `cor_select()`, `cor_df()`, and `cor_cramer_v()` for further details.
- **VIF filtering**: Automated algorithm to identify and remove numeric predictors that are linear combinations of other predictors. See `vif_select()` and `vif_df()`.

Accepts a parallelization setup via `future::plan()` and a progress bar via `progressr::handlers()` (see examples).

Accepts a character vector of response variables as input for the argument `response`. When more than one response is provided, the output is a named list of character.

Usage

```
collinear(
  df = NULL,
  response = NULL,
  predictors = NULL,
  encoding_method = "loo",
  preference_order = "auto",
  f = "auto",
  max_cor = 0.75,
  max_vif = 5,
  quiet = FALSE
)
```

Arguments

`df`	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
`response`	(optional; character string or vector) Name/s of response variable/s in `df`. Used in target encoding when it names a numeric variable and there are categorical predictors, and to compute preference order. Default: NULL.
`predictors`	(optional; character vector) Names of the predictors to select from `df`. If omitted, all numeric columns in `df` are used instead. If argument `response` is not provided, non-numeric variables are ignored. Default: NULL
`encoding_method`	(optional; character string). Name of the target encoding method. One of: "loo", "mean", or "rank". If NULL, target encoding is disabled. Default: "loo"
`preference_order`	(optional; string, character vector, output of `preference_order()`). Defines a priority order, from first to last, to preserve predictors during the selection process. Accepted inputs are:

- **"auto"** (default): if `response` is not NULL, calls `preference_order()` for internal computation.
- **character vector**: predictor names in a custom preference order.
- **data frame**: output of `preference_order()` from `response` of length one.
- **named list**: output of `preference_order()` from `response` of length two or more.
- **NULL**: disabled.

. Default: "auto"

`f` (optional: function) Function to compute preference order. If "auto" (default) or NULL, the output of `f_auto()` for the given data is used:

- `f_auc_rf()`: if `response` is binomial.
- `f_r2_pearson()`: if `response` and `predictors` are numeric.
- `f_v()`: if `response` and `predictors` are categorical.
- `f_v_rf_categorical()`: if `response` is categorical and `predictors` are numeric or mixed .
- `f_r2_rf()`: in all other cases.

Default: NULL

`max_cor` (optional; numeric) Maximum correlation allowed between any pair of variables in `predictors`. Recommended values are between 0.5 and 0.9. Higher values return larger number of predictors with a higher multicollinearity. If NULL, the pairwise correlation analysis is disabled. Default: `0.75`

`max_vif` (optional, numeric) Maximum Variance Inflation Factor allowed during variable selection. Recommended values are between 2.5 and 10. Higher values return larger number of predictors with a higher multicollinearity. If NULL, the variance inflation analysis is disabled. Default: 5.

`quiet` (optional; logical) If FALSE, messages generated during the execution of the function are printed to the console Default: FALSE

Value

- character vector if `response` is NULL or is a string.
- named list if `response` is a character vector.

Target Encoding

When the argument `response` names a **numeric response variable**, categorical predictors in `predictors` (or in the columns of `df` if `predictors` is NULL) are converted to numeric via **target encoding** with the function `target_encoding_lab()`. When `response` is NULL or names a categorical variable, target-encoding is skipped. This feature facilitates multicollinearity filtering in data frames with mixed column types.

Preference Order

This feature is designed to help protect important predictors during the multicollinearity filtering. It involves the arguments `preference_order` and `f`.

The argument `preference_order` accepts:

- : A character vector of predictor names in a custom order of preference, from first to last. This vector does not need to contain all predictor names, but only the ones relevant to the user.
- A data frame returned by `preference_order()`, which ranks predictors based on their association with a response variable.
- If NULL, and `response` is provided, then `preference_order()` is used internally to rank the predictors using the function `f`. If `f` is NULL as well, then `f_auto()` selects a proper function based on the data properties.

Variance Inflation Factors

The Variance Inflation Factor for a given variable a is computed as $1/(1 - R2)$, where $R2$ is the multiple R-squared of a multiple regression model fitted using a as response and all other predictors in the input data frame as predictors, as in $a = b + c + ...$

The square root of the VIF of a is the factor by which the confidence interval of the estimate for a in the linear model $y = a + b + c + ...$‘ is widened by multicollinearity in the model predictors.

The range of VIF values is (1, Inf]. The recommended thresholds for maximum VIF may vary depending on the source consulted, being the most common values, 2.5, 5, and 10.

VIF-based Filtering

The function `vif_select()` computes Variance Inflation Factors and removes variables iteratively, until all variables in the resulting selection have a VIF below `max_vif`.

If the argument `preference_order` is not provided, all variables are ranked from lower to higher VIF, as returned by `vif_df()`, and the variable with the higher VIF above `max_vif` is removed on each iteration.

If `preference_order` is defined, whenever two or more variables are above `max_vif`, the one higher in `preference_order` is preserved, and the next one with a higher VIF is removed. For example, for the predictors and preference order a and b, if any of their VIFs is higher than `max_vif`, then b will be removed no matter whether its VIF is lower or higher than a's VIF. If their VIF scores are lower than `max_vif`, then both are preserved.

Pairwise Correlation Filtering

The function `cor_select()` applies a recursive forward selection algorithm to keep predictors with a maximum Pearson correlation with all other selected predictors lower than `max_cor`.

If the argument `preference_order` is NULL, the predictors are ranked from lower to higher sum of absolute pairwise correlation with all other predictors.

If `preference_order` is defined, whenever two or more variables are above `max_cor`, the one higher in `preference_order` is preserved. For example, for the predictors and preference order a and b, if their correlation is higher than `max_cor`, then b will be removed and a preserved. If their correlation is lower than `max_cor`, then both are preserved.

References

- David A. Belsley, D.A., Kuh, E., Welsch, R.E. (1980). Regression Diagnostics: Identifying Influential Data and Sources of Collinearity. John Wiley & Sons. DOI: 10.1002/0471725153.

- Micci-Barreca, D. (2001) A Preprocessing Scheme for High-Cardinality Categorical Attributes in Classification and Prediction Problems. SIGKDD Explor. Newsl. 3, 1, 27-32. DOI: 10.1145/507533.507538

Examples

```
#parallelization setup
future::plan(
  future::multisession,
  workers = 2 #set to parallelly::availableCores() - 1
)

#progress bar
#progressr::handlers(global = TRUE)

#subset to limit example run time
df <- vi[1:500, ]

#predictors has mixed types
#small subset to speed example up
predictors <- c(
  "swi_mean",
  "soil_type",
  "soil_temperature_mean",
  "growing_season_length",
  "rainfall_mean"
  )


#with numeric responses
#-------------------------------
#  target encoding
#  automated preference order
#  all predictors filtered by correlation and VIF
x <- collinear(
  df = df,
  response = c(
    "vi_numeric",
    "vi_binomial"
    ),
  predictors = predictors
)

x


#with custom preference order
#-------------------------------
x <- collinear(
  df = df,
  response = "vi_numeric",
  predictors = predictors,
  preference_order = c(
```

```
    "swi_mean",
    "soil_type"
  )
)


#pre-computed preference order
#-------------------------------
preference_df <- preference_order(
  df = df,
  response = "vi_numeric",
  predictors = predictors
)

x <- collinear(
  df = df,
  response = "vi_numeric",
  predictors = predictors,
  preference_order = preference_df
)

#resetting to sequential processing
future::plan(future::sequential)
```

`cor_clusters` *Hierarchical Clustering from a Pairwise Correlation Matrix*

Description

Hierarchical clustering of predictors from their pairwise correlation matrix. Computes the correlation matrix with `cor_df()` and `cor_matrix()`, transforms it to a dist object, computes a clustering solution with `stats::hclust()`, and applies `stats::cutree()` to separate groups based on the value of the argument `max_cor`.

Returns a data frame with predictor names and their clusters, and optionally, prints a dendrogram of the clustering solution.

Accepts a parallelization setup via `future::plan()` and a progress bar via `progressr::handlers()` (see examples).

Usage

```
cor_clusters(
  df = NULL,
  predictors = NULL,
  max_cor = 0.75,
  method = "complete",
  plot = FALSE
)
```

Arguments

df	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
predictors	(optional; character vector) Names of the predictors to select from df. If omitted, all numeric columns in df are used instead. If argument response is not provided, non-numeric variables are ignored. Default: NULL
max_cor	(optional; numeric) Maximum correlation allowed between any pair of variables in predictors. Recommended values are between 0.5 and 0.9. Higher values return larger number of predictors with a higher multicollinearity. If NULL, the pairwise correlation analysis is disabled. Default: 0.75
method	(optional, character string) Argument of stats::hclust() defining the agglomerative method. One of: "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC). Unambiguous abbreviations are accepted as well. Default: "complete".
plot	(optional, logical) If TRUE, the clustering is plotted. Default: FALSE

Value

data frame: predictor names and their clusters

See Also

Other pairwise_correlation: cor_cramer_v(), cor_df(), cor_matrix(), cor_select()

Examples

```
#parallelization setup
future::plan(
  future::multisession,
  workers = 2 #set to parallelly::availableCores() - 1
)

#progress bar
# progressr::handlers(global = TRUE)

df_clusters <- cor_clusters(
  df = vi[1:1000, ],
  predictors = vi_predictors[1:15]
)

#disable parallelization
future::plan(future::sequential)
```

`cor_cramer_v` *Bias Corrected Cramer's V*

Description

Computes bias-corrected Cramer's V (extension of the chi-squared test), a measure of association between two categorical variables. Results are in the range 0-1, where 0 indicates no association, and 1 indicates a perfect association.

In essence, Cramer's V assesses the co-occurrence of the categories of two variables to quantify how strongly these variables are related.

Even when its range is between 0 and 1, Cramer's V values are not directly comparable to R-squared values, and as such, a multicollinearity analysis containing both types of values must be assessed with care. It is probably preferable to convert non-numeric variables to numeric using target encoding rather before a multicollinearity analysis.

Usage

```
cor_cramer_v(x = NULL, y = NULL, check_input = TRUE)
```

Arguments

`x`	(required; character vector) character vector representing a categorical variable. Default: NULL
`y`	(required; character vector) character vector representing a categorical variable. Must have the same length as 'x'. Default: NULL
`check_input`	(required; logical) If FALSE, disables data checking for a slightly faster execution. Default: TRUE

Value

numeric: Cramer's V

Author(s)

Blas M. Benito, PhD

References

- Cramér, H. (1946). Mathematical Methods of Statistics. Princeton: Princeton University Press, page 282 (Chapter 21. The two-dimensional case). ISBN 0-691-08004-6

See Also

Other pairwise_correlation: `cor_clusters()`, `cor_df()`, `cor_matrix()`, `cor_select()`

Examples

```
#loading example data
data(vi)

#subset to limit example run time
vi <- vi[1:1000, ]

#computing Cramer's V for two categorical predictors
v <- cor_cramer_v(
  x = vi$soil_type,
  y = vi$koppen_zone
  )

v
```

---

cor_df *Pairwise Correlation Data Frame*

Description

Computes a pairwise correlation data frame. Implements methods to compare different types of predictors:

- **numeric vs. numeric**: as computed with `stats::cor()` using the methods "pearson" or "spearman", via `cor_numeric_vs_numeric()`.
- **numeric vs. categorical**: the function `cor_numeric_vs_categorical()` target-encodes the categorical variable using the numeric variable as reference with `target_encoding_lab()` and the method "loo" (leave-one-out), and then their correlation is computed with `stats::cor()`.
- **categorical vs. categorical**: the function `cor_categorical_vs_categorical()` computes Cramer's V (see `cor_cramer_v()`) as indicator of the association between character or factor variables. However, take in mind that Cramer's V is not directly comparable with R-squared, even when having the same range from zero to one. It is always recommended to target-encode categorical variables with `target_encoding_lab()` before the pairwise correlation analysis.

Accepts a parallelization setup via `future::plan()` and a progress bar via `progressr::handlers()` (see examples).

Usage

```
cor_df(df = NULL, predictors = NULL, quiet = FALSE)

cor_numeric_vs_numeric(df = NULL, predictors = NULL, quiet = FALSE)

cor_numeric_vs_categorical(df = NULL, predictors = NULL, quiet = FALSE)

cor_categorical_vs_categorical(df = NULL, predictors = NULL, quiet = FALSE)
```

Arguments

df	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
predictors	(optional; character vector) Names of the predictors to select from df. If omitted, all numeric columns in df are used instead. If argument response is not provided, non-numeric variables are ignored. Default: NULL
quiet	(optional; logical) If FALSE, messages generated during the execution of the function are printed to the console Default: FALSE

Value

data frame; pairwise correlation

See Also

Other pairwise_correlation: cor_clusters(), cor_cramer_v(), cor_matrix(), cor_select()

Other pairwise_correlation: cor_clusters(), cor_cramer_v(), cor_matrix(), cor_select()

Other pairwise_correlation: cor_clusters(), cor_cramer_v(), cor_matrix(), cor_select()

Other pairwise_correlation: cor_clusters(), cor_cramer_v(), cor_matrix(), cor_select()

Examples

```
data(
  vi,
  vi_predictors
)

#reduce size of vi to speed-up example execution
vi <- vi[1:1000, ]

#mixed predictors
vi_predictors <- vi_predictors[1:10]

#parallelization setup
future::plan(
  future::multisession,
  workers = 2 #set to parallelly::availableCores() - 1
)

#progress bar
# progressr::handlers(global = TRUE)

#correlation data frame
df <- cor_df(
  df = vi,
  predictors = vi_predictors
)

df
```

```
#disable parallelization
future::plan(future::sequential)
```

`cor_matrix` *Pairwise Correlation Matrix*

Description

If argument 'df' results from `cor_df()`, transforms it to a correlation matrix. If argument 'df' is a dataframe with predictors, and the argument 'predictors' is provided then `cor_df()` is used to compute pairwise correlations, and the result is transformed to matrix.

Accepts a parallelization setup via `future::plan()` and a progress bar via `progressr::handlers()` (see examples).

Usage

```
cor_matrix(df = NULL, predictors = NULL)
```

Arguments

`df`	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
`predictors`	(optional; character vector) Names of the predictors to select from `df`. If omitted, all numeric columns in `df` are used instead. If argument `response` is not provided, non-numeric variables are ignored. Default: NULL

Value

correlation matrix

Author(s)

Blas M. Benito, PhD

See Also

Other pairwise_correlation: `cor_clusters()`, `cor_cramer_v()`, `cor_df()`, `cor_select()`

Examples

```
data(
  vi,
  vi_predictors
)

#reduce size of vi to speed-up example execution
```

```
vi <- vi[1:1000, ]

#mixed predictors
vi_predictors <- vi_predictors[1:10]

#parallelization setup
future::plan(
  future::multisession,
  workers = 2 #set to parallelly::availableCores() - 1
)

#progress bar
# progressr::handlers(global = TRUE)

#correlation data frame
df <- cor_df(
  df = vi,
  predictors = vi_predictors
)

df

#correlation matrix
m <- cor_matrix(
  df = df
)

m

#generating it from the original data
m <- cor_matrix(
  df = vi,
  predictors = vi_predictors
)

m

#disable parallelization
future::plan(future::sequential)
```

`cor_select` *Automated Multicollinearity Filtering with Pairwise Correlations*

Description

Implements a recursive forward selection algorithm to keep predictors with a maximum pairwise correlation with all other selected predictors lower than a given threshold. Uses `cor_df()` underneath, and as such, can handle different combinations of predictor types.

Please check the section **Pairwise Correlation Filtering** at the end of this help file for further details.

Usage

```
cor_select(
  df = NULL,
  predictors = NULL,
  preference_order = NULL,
  max_cor = 0.75,
  quiet = FALSE
)
```

Arguments

`df`	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
`predictors`	(optional; character vector) Names of the predictors to select from `df`. If omitted, all numeric columns in `df` are used instead. If argument `response` is not provided, non-numeric variables are ignored. Default: NULL
`preference_order`	(optional; string, character vector, output of `preference_order()`). Defines a priority order, from first to last, to preserve predictors during the selection process. Accepted inputs are: • **"auto"** (default): if `response` is not NULL, calls `preference_order()` for internal computation. • **character vector**: predictor names in a custom preference order. • **data frame**: output of `preference_order()` from `response` of length one. • **named list**: output of `preference_order()` from `response` of length two or more. • **NULL**: disabled. . Default: "auto"
`max_cor`	(optional; numeric) Maximum correlation allowed between any pair of variables in `predictors`. Recommended values are between 0.5 and 0.9. Higher values return larger number of predictors with a higher multicollinearity. If NULL, the pairwise correlation analysis is disabled. Default: `0.75`
`quiet`	(optional; logical) If FALSE, messages generated during the execution of the function are printed to the console Default: FALSE

Value

- character vector if `response` is NULL or is a string.
- named list if `response` is a character vector.

Pairwise Correlation Filtering

The function `cor_select()` applies a recursive forward selection algorithm to keep predictors with a maximum Pearson correlation with all other selected predictors lower than `max_cor`.

If the argument `preference_order` is NULL, the predictors are ranked from lower to higher sum of absolute pairwise correlation with all other predictors.

If preference_order is defined, whenever two or more variables are above max_cor, the one higher in preference_order is preserved. For example, for the predictors and preference order a and b, if their correlation is higher than max_cor, then b will be removed and a preserved. If their correlation is lower than max_cor, then both are preserved.

Author(s)

Blas M. Benito, PhD

See Also

Other pairwise_correlation: cor_clusters(), cor_cramer_v(), cor_df(), cor_matrix()

Examples

```
#subset to limit example run time
df <- vi[1:1000, ]

#only numeric predictors only to speed-up examples
#categorical predictors are supported, but result in a slower analysis
predictors <- vi_predictors_numeric[1:8]

#predictors has mixed types
sapply(
  X = df[, predictors, drop = FALSE],
  FUN = class
)

#parallelization setup
future::plan(
  future::multisession,
  workers = 2 #set to parallelly::availableCores() - 1
)

#progress bar
# progressr::handlers(global = TRUE)

#without preference order
x <- cor_select(
  df = df,
  predictors = predictors,
  max_cor = 0.75
)


#with custom preference order
x <- cor_select(
  df = df,
  predictors = predictors,
  preference_order = c(
    "swi_mean",
    "soil_type"
```

```
  ),
  max_cor = 0.75
)


#with automated preference order
df_preference <- preference_order(
  df = df,
  response = "vi_numeric",
  predictors = predictors
)

x <- cor_select(
  df = df,
  predictors = predictors,
  preference_order = df_preference,
  max_cor = 0.75
)

#resetting to sequential processing
future::plan(future::sequential)
```

`drop_geometry_column` *Removes geometry column in sf data frames*

Description

Replicates the functionality of `sf::st_drop_geometry()` without depending on the `sf` package.

Usage

```
drop_geometry_column(df = NULL, quiet = FALSE)
```

Arguments

`df`	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
`quiet`	(optional; logical) If FALSE, messages generated during the execution of the function are printed to the console Default: FALSE

Value

data frame

Author(s)

Blas M. Benito, PhD

encoded_predictor_name *Name of Target-Encoded Predictor*

Description

Name of Target-Encoded Predictor

Usage

```
encoded_predictor_name(
  predictor = NULL,
  encoding_method = "mean",
  smoothing = 0,
  white_noise = 0,
  seed = 1
)
```

Arguments

predictor	(required; string) Name of the categorical predictor to encode. Default: NULL
encoding_method	(required, string) Name of the encoding method. One of: "mean", "rank", or "loo". Default: "mean"
smoothing	(optional; integer) Groups smaller than this number have their means pulled towards the mean of the response across all cases. Ignored by `target_encoding_rank()` and `target_encoding_loo()`. Default: 0
white_noise	(optional; numeric vector) Argument of the methods "mean", "rank", and "loo". Maximum white noise to add, expressed as a fraction of the range of the response variable. Range from 0 to 1. Default: `0`.
seed	(optional; integer vector) Random seed to facilitate reproducibility when `white_noise` is not 0. If NULL, the function selects one at random, and the selected seed does not appear in the encoded variable names. Default: 0

Value

string: predictor name

See Also

Other target_encoding_tools: `add_white_noise()`

f_auc *Association Between a Binomial Response and a Continuous Predictor*

Description

These functions take a data frame with a binomial response "y" with unique values 1 and 0, and a continuous predictor "x", fit a univariate model, to return the Area Under the ROC Curve (AUC) of observations versus predictions:

- `f_auc_glm_binomial()`: AUC of a binomial response against the predictions of a GLM model with formula `y ~ x`, family `stats::quasibinomial(link = "logit")`, and weighted cases (see `case_weights()`) to control for unbalanced data.
- `f_auc_glm_binomial_poly2()`: AUC of a binomial response against the predictions of a GLM model with formula `y ~ stats::poly(x, degree = 2, raw = TRUE)`, family `stats::quasibinomial(link = "logit")`, and weighted cases (see `case_weights()`) to control for unbalanced data.
- `f_auc_gam_binomial()`: AUC of a GAM model with formula `y ~ s(x)`, family `stats::quasibinomial(link = "logit")`, and weighted cases.
- `f_auc_rpart()`: AUC of a Recursive Partition Tree with weighted cases.
- `f_auc_rf()`: AUC of a Random Forest model with weighted cases.

Usage

```
f_auc_glm_binomial(df)

f_auc_glm_binomial_poly2(df)

f_auc_gam_binomial(df)

f_auc_rpart(df)

f_auc_rf(df)
```

Arguments

`df`	(required, data frame) with columns: • "x": (numeric) continuous predictor. • "y" (integer) binomial response with unique values 0 and 1.

See Also

Other preference_order_functions: `f_r2`, `f_r2_counts`, `f_v()`, `f_v_rf_categorical()`

Other preference_order_functions: `f_r2`, `f_r2_counts`, `f_v()`, `f_v_rf_categorical()`

Other preference_order_functions: `f_r2`, `f_r2_counts`, `f_v()`, `f_v_rf_categorical()`

Other preference_order_functions: `f_r2`, `f_r2_counts`, `f_v()`, `f_v_rf_categorical()`

Other preference_order_functions: f_r2, f_r2_counts, f_v(), f_v_rf_categorical()

Other preference_order_functions: f_r2, f_r2_counts, f_v(), f_v_rf_categorical()

Examples

```
#load example data
data(vi)

#reduce size to speed-up example
vi <- vi[1:1000, ]

#integer counts response and continuous predictor
#to data frame without NAs
df <- data.frame(
  y = vi[["vi_binomial"]],
  x = vi[["swi_max"]]
) |>
  na.omit()

#AUC of GLM with binomial response and weighted cases
f_auc_glm_binomial(df = df)

#AUC of GLM as above plus second degree polynomials
f_auc_glm_binomial_poly2(df = df)

#AUC of binomial GAM with weighted cases
f_auc_gam_binomial(df = df)

#AUC of recursive partition tree with weighted cases
f_auc_rpart(df = df)

#AUC of random forest with weighted cases
f_auc_rf(df = df)
```

---

f_auto *Select Function to Compute Preference Order*

Description

Internal function to select a proper f_...() function to compute preference order depending on the types of the response variable and the predictors. The selection criteria is available as a data frame generated by f_auto_rules().

Usage

```
f_auto(df = NULL, response = NULL, predictors = NULL, quiet = FALSE)
```

Arguments

df	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
response	(optional; character string or vector) Name/s of response variable/s in df. Used in target encoding when it names a numeric variable and there are categorical predictors, and to compute preference order. Default: NULL.
predictors	(optional; character vector) Names of the predictors to select from df. If omitted, all numeric columns in df are used instead. If argument response is not provided, non-numeric variables are ignored. Default: NULL
quiet	(optional; logical) If FALSE, messages generated during the execution of the function are printed to the console Default: FALSE

Value

function name

See Also

Other preference_order_tools: f_auto_rules(), f_functions(), preference_order_collinear()

Examples

```
f <- f_auto(
  df = vi[1:1000, ],
  response = "vi_numeric",
  predictors = vi_predictors_numeric
  )
```

---

f_auto_rules *Rules to Select Default f Argument to Compute Preference Order*

Description

Data frame with rules used by f_auto() to select the function f to compute preference order in preference_order().

Usage

```
f_auto_rules()
```

Value

data frame

See Also

Other preference_order_tools: f_auto(), f_functions(), preference_order_collinear()

Examples

```
f_auto_rules()
```

`f_functions` *Data Frame of Preference Functions*

Description

Data Frame of Preference Functions

Usage

```
f_functions()
```

Value

data frame

See Also

Other preference_order_tools: `f_auto()`, `f_auto_rules()`, `preference_order_collinear()`

Examples

```
f_functions()
```

`f_r2` *Association Between a Continuous Response and a Continuous Predictor*

Description

These functions take a data frame with two numeric continuous columns "x" (predictor) and "y" (response), fit a univariate model, and return the R-squared of the observations versus the model predictions:

- `f_r2_pearson()`: Pearson's R-squared.
- `f_r2_spearman()`: Spearman's R-squared.
- `f_r2_glm_gaussian()`: Pearson's R-squared of a GLM model fitted with `stats::glm()`, with formula `y ~ s(x)` and family `stats::gaussian(link = "identity")`.
- `f_r2_glm_gaussian_poly2()`: Pearson's R-squared of a GLM model fitted with `stats::glm()`, with formula `y ~ stats::poly(x, degree = 2, raw = TRUE)` and family `stats::gaussian(link = "identity")`.
- `f_r2_gam_gaussian()`: Pearson's R-squared of a GAM model fitted with `mgcv::gam()`, with formula `y ~ s(x)` and family `stats::gaussian(link = "identity")`.

- `f_r2_rpart()`: Pearson's R-squared of a Recursive Partition Tree fitted with `rpart::rpart()` with formula `y ~ x`.
- `f_r2_rf()`: Pearson's R-squared of a 100 trees Random Forest model fitted with `ranger::ranger()` and formula `y ~ x`.

Usage

```
f_r2_pearson(df)

f_r2_spearman(df)

f_r2_glm_gaussian(df)

f_r2_glm_gaussian_poly2(df)

f_r2_gam_gaussian(df)

f_r2_rpart(df)

f_r2_rf(df)
```

Arguments

`df`	(required, data frame) with columns: • "x": (numeric) continuous predictor. • "y" (numeric) continuous response.

Value

numeric: R-squared

See Also

Other preference_order_functions: `f_auc`, `f_r2_counts`, `f_v()`, `f_v_rf_categorical()`

Other preference_order_functions: `f_auc`, `f_r2_counts`, `f_v()`, `f_v_rf_categorical()`

Other preference_order_functions: `f_auc`, `f_r2_counts`, `f_v()`, `f_v_rf_categorical()`

Other preference_order_functions: `f_auc`, `f_r2_counts`, `f_v()`, `f_v_rf_categorical()`

Other preference_order_functions: `f_auc`, `f_r2_counts`, `f_v()`, `f_v_rf_categorical()`

Other preference_order_functions: `f_auc`, `f_r2_counts`, `f_v()`, `f_v_rf_categorical()`

Other preference_order_functions: `f_auc`, `f_r2_counts`, `f_v()`, `f_v_rf_categorical()`

Other preference_order_functions: `f_auc`, `f_r2_counts`, `f_v()`, `f_v_rf_categorical()`

Examples

```
data(vi)

#reduce size to speed-up example
vi <- vi[1:1000, ]

#numeric response and predictor
#to data frame without NAs
df <- data.frame(
  y = vi[["vi_numeric"]],
  x = vi[["swi_max"]]
) |>
  na.omit()

# Continuous response

#Pearson R-squared
f_r2_pearson(df = df)

#Spearman R-squared
f_r2_spearman(df = df)

#R-squared of a gaussian gam
f_r2_glm_gaussian(df = df)

#gaussian glm with second-degree polynomials
f_r2_glm_gaussian_poly2(df = df)

#R-squared of a gaussian gam
f_r2_gam_gaussian(df = df)

#recursive partition tree
f_r2_rpart(df = df)

#random forest model
f_r2_rf(df = df)

#load example data
data(vi)

#reduce size to speed-up example
vi <- vi[1:1000, ]

#continuous response and predictor
#to data frame without NAs
df <- data.frame(
  y = vi[["vi_numeric"]],
  x = vi[["swi_max"]]
) |>
  na.omit()

# Continuous response
```

```
#Pearson R-squared
f_r2_pearson(df = df)

#Spearman R-squared
f_r2_spearman(df = df)

#R-squared of a gaussian gam
f_r2_glm_gaussian(df = df)

#gaussian glm with second-degree polynomials
f_r2_glm_gaussian_poly2(df = df)

#R-squared of a gaussian gam
f_r2_gam_gaussian(df = df)

#recursive partition tree
f_r2_rpart(df = df)

#random forest model
f_r2_rf(df = df)
```

`f_r2_counts` *Association Between a Count Response and a Continuous Predictor*

Description

These functions take a data frame with a integer counts response "y", and a continuous predictor "x", fit a univariate model, and return the R-squared of observations versus predictions:

- `f_r2_glm_poisson()` Pearson's R-squared between a count response and the predictions of a GLM model with formula `y ~ x` and family `stats::poisson(link = "log")`.
- `f_r2_glm_poisson_poly2()` Pearson's R-squared between a count response and the predictions of a GLM model with formula `y ~ stats::poly(x, degree = 2, raw = TRUE)` and family `stats::poisson(link = "log")`.
- `f_r2_gam_poisson()` Pearson's R-squared between a count response and the predictions of a `mgcv::gam()` model with formula `y ~ s(x)` and family `stats::poisson(link = "log")`.
- `f_r2_rpart()`: Pearson's R-squared of a Recursive Partition Tree fitted with `rpart::rpart()` with formula `y ~ x`.
- `f_r2_rf()`: Pearson's R-squared of a 100 trees Random Forest model fitted with `ranger::ranger()` and formula `y ~ x`.

Usage

```
f_r2_glm_poisson(df)

f_r2_glm_poisson_poly2(df)

f_r2_gam_poisson(df)
```

Arguments

df	(required, data frame) with columns: • "x": (numeric) continuous predictor. • "y" (integer) counts response.

See Also

Other preference_order_functions: f_auc, f_r2, f_v(), f_v_rf_categorical()

Other preference_order_functions: f_auc, f_r2, f_v(), f_v_rf_categorical()

Other preference_order_functions: f_auc, f_r2, f_v(), f_v_rf_categorical()

Other preference_order_functions: f_auc, f_r2, f_v(), f_v_rf_categorical()

Examples

```
#load example data
data(vi)

#reduce size to speed-up example
vi <- vi[1:1000, ]

#integer counts response and continuous predictor
#to data frame without NAs
df <- data.frame(
  y = vi[["vi_counts"]],
  x = vi[["swi_max"]]
) |>
  na.omit()

#GLM model with Poisson family
f_r2_glm_poisson(df = df)

#GLM model with second degree polynomials and Poisson family
f_r2_glm_poisson_poly2(df = df)

#GAM model with Poisson family
f_r2_gam_poisson(df = df)
```

---

f_v *Association Between a Categorical Response and a Categorical Predictor*

Description

Computes Cramer's V, a measure of association between categorical or factor variables. Please see cor_cramer_v() for further details.

Usage

```
f_v(df)
```

Arguments

df	(required, data frame) with columns: • "x": (character or factor) categorical predictor. • "y": (character or factor) categorical response.

Value

numeric: Cramer's V

See Also

Other preference_order_functions: f_auc, f_r2, f_r2_counts, f_v_rf_categorical()

Examples

```
#load example data
data(vi)

#reduce size to speed-up example
vi <- vi[1:1000, ]

#categorical response and predictor
#to data frame without NAs
df <- data.frame(
  y = vi[["vi_factor"]],
  x = vi[["soil_type"]]
) |>
  na.omit()

#Cramer's V
f_v(df = df)
```

---

f_v_rf_categorical *Association Between a Categorical Response and a Categorical or Numeric Predictor*

Description

Computes the Cramer's V between a categorical response (of class "character" or "factor") and the prediction of a Random Forest model with a categorical or numeric predictor and weighted cases.

Usage

```
f_v_rf_categorical(df)
```

Arguments

df	(required, data frame) with columns: • "x": (character, factor, or numeric) categorical or numeric predictor. • "y" (character or factor) categorical response.

Value

numeric: Cramer's V

See Also

Other preference_order_functions: f_auc, f_r2, f_r2_counts, f_v()

Examples

```
#load example data
data(vi)

#reduce size to speed-up example
vi <- vi[1:1000, ]

#categorical response and predictor
#to data frame without NAs
df <- data.frame(
  y = vi[["vi_factor"]],
  x = vi[["soil_type"]]
) |>
  na.omit()

#Cramer's V of a Random Forest model
f_v_rf_categorical(df = df)

#categorical response and numeric predictor
df <- data.frame(
  y = vi[["vi_factor"]],
  x = vi[["swi_mean"]]
) |>
  na.omit()

f_v_rf_categorical(df = df)
```

---

identify_predictors *Identify Numeric and Categorical Predictors*

Description

Returns a list with the names of the valid numeric predictors and the names of the valid categorical predictors

Usage

```
identify_predictors(df = NULL, predictors = NULL)
```

Arguments

df	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
predictors	(optional; character vector) Names of the predictors to select from df. If omitted, all numeric columns in df are used instead. If argument response is not provided, non-numeric variables are ignored. Default: NULL

Value

list: names of numeric and categorical predictors

Author(s)

Blas M. Benito, PhD

See Also

Other data_types: identify_predictors_categorical(), identify_predictors_numeric(), identify_predictors_type(), identify_predictors_zero_variance(), identify_response_type()

Examples

```
if (interactive()) {

data(
  vi,
  vi_predictors
)

predictors_names <- identify_predictors(
  df = vi,
  predictors = vi_predictors
)

predictors_names

}
```

identify_predictors_categorical
Identify Valid Categorical Predictors

Description

Returns the names of character or factor predictors, if any. Removes categorical predictors with constant values, or with as many unique values as rows are in the input data frame.

Usage

```
identify_predictors_categorical(df = NULL, predictors = NULL)
```

Arguments

df	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
predictors	(optional; character vector) Names of the predictors to select from df. If omitted, all numeric columns in df are used instead. If argument response is not provided, non-numeric variables are ignored. Default: NULL

Value

character vector: categorical predictors names

Author(s)

Blas M. Benito, PhD

See Also

Other data_types: identify_predictors(), identify_predictors_numeric(), identify_predictors_type(), identify_predictors_zero_variance(), identify_response_type()

Examples

```
data(
  vi,
  vi_predictors
)

non.numeric.predictors <- identify_predictors_categorical(
  df = vi,
  predictors = vi_predictors
)

non.numeric.predictors
```

identify_predictors_numeric *Identify Valid Numeric Predictors*

Description

Returns the names of valid numeric predictors. Ignores predictors with constant values or with near-zero variance.

Usage

```
identify_predictors_numeric(df = NULL, predictors = NULL, decimals = 4)
```

Arguments

df	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
predictors	(optional; character vector) Names of the predictors to select from df. If omitted, all numeric columns in df are used instead. If argument response is not provided, non-numeric variables are ignored. Default: NULL
decimals	(required, integer) Number of decimal places for the zero variance test. Smaller numbers will increase the number of variables detected as near-zero variance. Recommended values will depend on the range of the numeric variables in 'df'. Default: 4

Value

character vector: names of numeric predictors

Author(s)

Blas M. Benito, PhD

See Also

Other data_types: identify_predictors(), identify_predictors_categorical(), identify_predictors_type(), identify_predictors_zero_variance(), identify_response_type()

Examples

```
if (interactive()) {

data(
  vi,
  vi_predictors
)

numeric.predictors <- identify_predictors_numeric(
```

```
  df = vi,
  predictors = vi_predictors
)

numeric.predictors

}
```

`identify_predictors_type` *Identify Predictor Types*

Description

Internal function to identify predictor types. The supported types are:

- "numeric": all predictors belong to the classes "numeric" and/or "integer".
- "categorical": all predictors belong to the classes "character" and/or "factor".
- "mixed": predictors are of types "numeric" and "categorical".
- "unknown": predictors of unknown type.

Usage

```
identify_predictors_type(df = NULL, predictors = NULL)
```

Arguments

`df`	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
`predictors`	(optional; character vector) Names of the predictors to select from `df`. If omitted, all numeric columns in `df` are used instead. If argument `response` is not provided, non-numeric variables are ignored. Default: NULL

Value

character string: predictors type

See Also

Other data_types: `identify_predictors()`, `identify_predictors_categorical()`, `identify_predictors_numeric()`
`identify_predictors_zero_variance()`, `identify_response_type()`

Examples

```
identify_predictors_type(
  df = vi,
  predictors = vi_predictors
)

identify_predictors_type(
  df = vi,
  predictors = vi_predictors_numeric
)

identify_predictors_type(
  df = vi,
  predictors = vi_predictors_categorical
)
```

`identify_predictors_zero_variance`

Identify Zero and Near-Zero Variance Predictors

Description

Variables with a variance of zero or near-zero are highly problematic for multicollinearity analysis and modelling in general. This function identifies these variables with a level of sensitivity defined by the 'decimals' argument. Smaller number of decimals increase the number of variables detected as near zero variance. Recommended values will depend on the range of the numeric variables in 'df'.

Usage

```
identify_predictors_zero_variance(df = NULL, predictors = NULL, decimals = 4)
```

Arguments

`df`	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
`predictors`	(optional; character vector) Names of the predictors to select from `df`. If omitted, all numeric columns in `df` are used instead. If argument `response` is not provided, non-numeric variables are ignored. Default: NULL
`decimals`	(required, integer) Number of decimal places for the zero variance test. Smaller numbers will increase the number of variables detected as near-zero variance. Recommended values will depend on the range of the numeric variables in 'df'. Default: 4

Value

character vector: names of zero and near-zero variance columns.

Author(s)

Blas M. Benito, PhD

See Also

Other data_types: identify_predictors(), identify_predictors_categorical(), identify_predictors_numeric()
identify_predictors_type(), identify_response_type()

Examples

```
data(
  vi,
  vi_predictors
)

#create zero variance predictors
vi$zv_1 <- 1
vi$zv_2 <- runif(n = nrow(vi), min = 0, max = 0.0001)


#add to vi predictors
vi_predictors <- c(
  vi_predictors,
  "zv_1",
  "zv_2"
)

#identify zero variance predictors
zero.variance.predictors <- identify_predictors_zero_variance(
  df = vi,
  predictors = vi_predictors
)

zero.variance.predictors
```

---

identify_response_type

Identify Response Type

Description

Internal function to identify the type of response variable. Supported types are:

- "continuous-binary": decimal numbers and two unique values; results in a warning, as this type is difficult to model.
- "continuous-low": decimal numbers and 3 to 5 unique values; results in a message, as this type is difficult to model.

- "continuous-high": decimal numbers and more than 5 unique values.
- "integer-binomial": integer with 0s and 1s, suitable for binomial models.
- "integer-binary": integer with 2 unique values other than 0 and 1; returns a warning, as this type is difficult to model.
- "integer-low": integer with 3 to 5 unique values or meets specified thresholds.
- "integer-high": integer with more than 5 unique values suitable for count modelling.
- "categorical": character or factor with 2 or more levels.
- "unknown": when the response type cannot be determined.

Usage

```
identify_response_type(df = NULL, response = NULL, quiet = FALSE)
```

Arguments

df	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
response	(optional; character string or vector) Name/s of response variable/s in df. Used in target encoding when it names a numeric variable and there are categorical predictors, and to compute preference order. Default: NULL.
quiet	(optional; logical) If FALSE, messages generated during the execution of the function are printed to the console Default: FALSE

Value

character string: response type

See Also

Other data_types: identify_predictors(), identify_predictors_categorical(), identify_predictors_numeric()
identify_predictors_type(), identify_predictors_zero_variance()

Examples

```
identify_response_type(
  df = vi,
  response = "vi_numeric"
)

identify_response_type(
  df = vi,
  response = "vi_counts"
)

identify_response_type(
  df = vi,
  response = "vi_binomial"
)
```

```
identify_response_type(
  df = vi,
  response = "vi_categorical"
)

identify_response_type(
  df = vi,
  response = "vi_factor"
)
```

`model_formula` *Generate Model Formulas*

Description

Generate Model Formulas

Usage

```
model_formula(
  df = NULL,
  response = NULL,
  predictors = NULL,
  term_f = NULL,
  term_args = NULL,
  random_effects = NULL,
  quiet = FALSE
)
```

Arguments

`df`	(optional; data frame, tibble, or sf). A data frame with responses and predictors. Required if `predictors = NULL`. Default: NULL.
`response`	(optional; character string or vector) Name/s of response variable/s in `df`. Used in target encoding when it names a numeric variable and there are categorical predictors, and to compute preference order. Default: NULL.
`predictors`	(optional, character vector, output of `collinear()`): predictors to include in the formula. Required if `df = NULL`.
`term_f`	(optional; string). Name of function to apply to each term in the formula, such as "s" for `mgcv::s()` or any other smoothing function, "poly" for `stats::poly()`. Default: NULL
`term_args`	(optional; string). Arguments of the function applied to each term. For example, for "poly" it can be "degree = 2, raw = TRUE". Default: NULL

random_effects	(optional, string or character vector). Names of variables to be used as random effects. Each element is added to the final formula as +(1 | random_effect_name). Default: NULL
quiet	(optional; logical) If FALSE, messages generated during the execution of the function are printed to the console Default: FALSE

Value

list if predictors is a list or length of response is higher than one, and character vector otherwise.

See Also

Other modelling_tools: case_weights(), performance_score_auc(), performance_score_r2(), performance_score_v()

Examples

```
#using df, response, and predictors
#----------------------------------
df <- vi[1:1000, ]

#additive formulas
formulas_additive <- model_formula(
  df = df,
  response = c(
    "vi_numeric",
    "vi_categorical"
    ),
  predictors = vi_predictors_numeric[1:10]
)

formulas_additive

#using a formula in a model
#m <- stats::lm(
#  formula = formulas_additive[[1]],
#  data = df
#  )

# using output of collinear()
#----------------------------------
selection <- collinear(
  df = df,
  response = c(
    "vi_numeric",
    "vi_binomial"
  ),
  predictors = vi_predictors_numeric[1:10],
  quiet = TRUE
)

#polynomial formulas
```

```
formulas_poly <- model_formula(
  predictors = selection,
  term_f = "poly",
  term_args = "degree = 3, raw = TRUE"
)

formulas_poly

#gam formulas
formulas_gam <- model_formula(
  predictors = selection,
  term_f = "s"
)

formulas_gam

#adding a random effect
formulas_random_effect <- model_formula(
  predictors = selection,
  random_effects = "country_name"
)

formulas_random_effect
```

`performance_score_auc` *Area Under the Curve of Binomial Observations vs Probabilistic Model Predictions*

Description

Internal function to compute the AUC of binomial models within `preference_order()`. As it is build for speed, this function does not check the inputs.

Usage

```
performance_score_auc(o = NULL, p = NULL)
```

Arguments

o	(required, binomial vector) Binomial response with values 0 and 1. Default: NULL
p	(required, numeric vector) Predictions of binomial model. Default: NULL

Value

numeric: Area Under the ROC Curve

See Also

Other modelling_tools: `case_weights()`, `model_formula()`, `performance_score_r2()`, `performance_score_v()`

performance_score_r2 *Pearson's R-squared of Observations vs Predictions*

Description

Internal function to compute the R-squared of observations versus model predictions.

Usage

```
performance_score_r2(o = NULL, p = NULL)
```

Arguments

o	(required, numeric vector) Response values. Default: NULL
p	(required, numeric vector) Model predictions. Default: NULL

Value

numeric: Pearson R-squared

See Also

Other modelling_tools: `case_weights()`, `model_formula()`, `performance_score_auc()`, `performance_score_v()`

performance_score_v *Cramer's V of Observations vs Predictions*

Description

Internal function to compute the Cramer's V of categorical observations versus categorical model predictions.

Usage

```
performance_score_v(o = NULL, p = NULL)
```

Arguments

o	(required, numeric vector) Response values. Default: NULL
p	(required, numeric vector) Model predictions. Default: NULL

Value

numeric: Cramer's V

See Also

Other modelling_tools: `case_weights()`, `model_formula()`, `performance_score_auc()`, `performance_score_r2()`

preference_order *Quantitative Variable Prioritization for Multicollinearity Filtering*

Description

Ranks a set of predictors by the strength of their association with a response. Aims to minimize the loss of important predictors during multicollinearity filtering.

The strength of association between the response and each predictor is computed by the function `f`. The `f` functions available are:

- **Numeric response vs numeric predictor**:
  - `f_r2_pearson()`: Pearson's R-squared.
  - `f_r2_spearman()`: Spearman's R-squared.
  - `f_r2_glm_gaussian()`: Pearson's R-squared of response versus the predictions of a Gaussian GLM.
  - `f_r2_glm_gaussian_poly2()`: Gaussian GLM with second degree polynomial.
  - `f_r2_gam_gaussian()`: GAM model fitted with `mgcv::gam()`.
  - `f_r2_rpart()`: Recursive Partition Tree fitted with `rpart::rpart()`.
  - `f_r2_rf()`: Random Forest model fitted with `ranger::ranger()`.
- **Integer counts response vs. numeric predictor**:
  - `f_r2_glm_poisson()`: Pearson's R-squared of a Poisson GLM.
  - `f_r2_glm_poisson_poly2()`: Poisson GLM with second degree polynomial.
  - `f_r2_gam_poisson()`: Poisson GAM.
- **Binomial response (1 and 0) vs. numeric predictor**:
  - `f_auc_glm_binomial()`: AUC of quasibinomial GLM with weighted cases.
  - `f_auc_glm_binomial_poly2()`: As above with second degree polynomial.
  - `f_auc_gam_binomial()`: Quasibinomial GAM with weighted cases.
  - `f_auc_rpart()`: Recursive Partition Tree with weighted cases.
  - `f_auc_rf()`: Random Forest model with weighted cases.
- **Categorical response (character of factor) vs. categorical predictor**:
  - `f_v()`: Cramer's V between two categorical variables.
- **Categorical response vs. categorical or numerical predictor**:
  - `f_v_rf_categorical()`: Cramer's V of a Random Forest model.

The name of the used function is stored in the attribute "f_name" of the output data frame. It can be retrieved via `attributes(df)$f_name`

Additionally, any custom function accepting a data frame with the columns "x" (predictor) and "y" (response) and returning a numeric indicator of association where higher numbers indicate higher association will work.

This function returns a data frame with the column "predictor", with predictor names ordered by the column "preference", with the result of `f`. This data frame, or the column "predictor" alone,

can be used as inputs for the argument `preference_order` in `collinear()`, `cor_select()`, and `vif_select()`.

Accepts a parallelization setup via `future::plan()` and a progress bar via `progressr::handlers()` (see examples).

Accepts a character vector of response variables as input for the argument `response`. When more than one response is provided, the output is a named list of preference data frames.

Usage

```
preference_order(
  df = NULL,
  response = NULL,
  predictors = NULL,
  f = "auto",
  warn_limit = NULL,
  quiet = FALSE
)
```

Arguments

`df`	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
`response`	(optional; character string or vector) Name/s of response variable/s in `df`. Used in target encoding when it names a numeric variable and there are categorical predictors, and to compute preference order. Default: NULL.
`predictors`	(optional; character vector) Names of the predictors to select from `df`. If omitted, all numeric columns in `df` are used instead. If argument `response` is not provided, non-numeric variables are ignored. Default: NULL
`f`	(optional: function) Function to compute preference order. If "auto" (default) or NULL, the output of `f_auto()` for the given data is used: • `f_auc_rf()`: if `response` is binomial. • `f_r2_pearson()`: if `response` and `predictors` are numeric. • `f_v()`: if `response` and `predictors` are categorical. • `f_v_rf_categorical()`: if `response` is categorical and `predictors` are numeric or mixed . • `f_r2_rf()`: in all other cases. Default: NULL
`warn_limit`	(optional, numeric) Preference value (R-squared, AUC, or Cramer's V) over which a warning flagging suspicious predictors is issued. Disabled if NULL. Default: NULL
`quiet`	(optional; logical) If FALSE, messages generated during the execution of the function are printed to the console Default: FALSE

Value

data frame: columns are "response", "predictor", "f" (function name), and "preference".

Author(s)

Blas M. Benito, PhD

Examples

```
#subsets to limit example run time
df <- vi[1:1000, ]
predictors <- vi_predictors[1:10]
predictors_numeric <- vi_predictors_numeric[1:10]

#parallelization setup
future::plan(
  future::multisession,
  workers = 2 #set to parallelly::availableCores() - 1
)

#progress bar
# progressr::handlers(global = TRUE)

#numeric response and predictors
#------------------------------------------------
#selects f automatically depending on data features
#applies f_r2_pearson() to compute correlation between response and predictors
df_preference <- preference_order(
  df = df,
  response = "vi_numeric",
  predictors = predictors_numeric,
  f = NULL
  )

#returns data frame ordered by preference
df_preference


#several responses
#------------------------------------------------
responses <- c(
  "vi_categorical",
  "vi_counts"
)

preference_list <- preference_order(
  df = df,
  response = responses,
  predictors = predictors
)

#returns a named list
names(preference_list)
preference_list[[1]]
preference_list[[2]]
```

```
#can be used in collinear()
# x <- collinear(
#   df = df,
#   response = responses,
#   predictors = predictors,
#   preference_order = preference_list
# )

#f function selected by user
#for binomial response and numeric predictors
# preference_order(
#   df = vi,
#   response = "vi_binomial",
#   predictors = predictors_numeric,
#   f = f_auc_glm_binomial
# )


#disable parallelization
future::plan(future::sequential)
```

`preference_order_collinear`

Preference Order Argument in collinear()

Description

Internal function to manage the argument `preference_order` in `collinear()`.

Usage

```
preference_order_collinear(
  df = NULL,
  response = NULL,
  predictors = NULL,
  preference_order = NULL,
  f = NULL,
  quiet = FALSE
)
```

Arguments

`df`	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
`response`	(optional; character string or vector) Name/s of response variable/s in `df`. Used in target encoding when it names a numeric variable and there are categorical predictors, and to compute preference order. Default: NULL.

predictors (optional; character vector) Names of the predictors to select from df. If omitted, all numeric columns in df are used instead. If argument response is not provided, non-numeric variables are ignored. Default: NULL

preference_order (optional; string, character vector, output of preference_order()). Defines a priority order, from first to last, to preserve predictors during the selection process. Accepted inputs are:

- **"auto"** (default): if response is not NULL, calls preference_order() for internal computation.
- **character vector**: predictor names in a custom preference order.
- **data frame**: output of preference_order() from response of length one.
- **named list**: output of preference_order() from response of length two or more.
- **NULL**: disabled.

. Default: "auto"

f (optional: function) Function to compute preference order. If "auto" (default) or NULL, the output of f_auto() for the given data is used:

- f_auc_rf(): if response is binomial.
- f_r2_pearson(): if response and predictors are numeric.
- f_v(): if response and predictors are categorical.
- f_v_rf_categorical(): if response is categorical and predictors are numeric or mixed .
- f_r2_rf(): in all other cases.

Default: NULL

quiet (optional; logical) If FALSE, messages generated during the execution of the function are printed to the console Default: FALSE

Value

character vector or NULL

See Also

Other preference_order_tools: f_auto(), f_auto_rules(), f_functions()

---

target_encoding_lab *Target Encoding Lab: Transform Categorical Variables to Numeric*

Description

Target encoding involves replacing the values of categorical variables with numeric ones derived from a "target variable", usually a model's response.

In essence, target encoding works as follows:

- 1. group all cases belonging to a unique value of the categorical variable.
- 2. compute a statistic of the target variable across the group cases.
- 3. assign the value of the statistic to the group.

The methods to compute the group statistic implemented here are:

- "mean" (implemented in `target_encoding_mean()`): Encodes categorical values with the group means of the response. Variables encoded with this method are identified with the suffix "__encoded_mean". It has a method to control overfitting implemented via the argument `smoothing`. The integer value of this argument indicates a threshold in number of rows. Groups above this threshold are encoded with the group mean, while groups below it are encoded with a weighted mean of the group's mean and the global mean. This method is named "mean smoothing" in the relevant literature.
- "rank" (implemented in `target_encoding_rank()`): Returns the rank of the group as a integer, being 1 he group with the lower mean of the response variable. Variables encoded with this method are identified with the suffix "__encoded_rank".
- "loo" (implemented in `target_encoding_loo()`): Known as the "leave-one-out method" in the literature, it encodes each categorical value with the mean of the response variable across all other group cases. This method controls overfitting better than "mean". Variables encoded with this method are identified with the suffix "__encoded_loo".

Accepts a parallelization setup via `future::plan()` and a progress bar via `progressr::handlers()` (see examples).

Usage

```
target_encoding_lab(
  df = NULL,
  response = NULL,
  predictors = NULL,
  methods = c("loo", "mean", "rank"),
  smoothing = 0,
  white_noise = 0,
  seed = 0,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

`df`	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.

response	(optional, character string) Name of a numeric response variable in df. Default: NULL.
predictors	(optional; character vector) Names of the predictors to select from df. If omitted, all numeric columns in df are used instead. If argument response is not provided, non-numeric variables are ignored. Default: NULL
methods	(optional; character vector or NULL). Name of the target encoding methods. If NULL, target encoding is ignored, and df is returned with no modification. Default: c("loo", "mean", "rank")
smoothing	(optional; integer vector) Argument of the method "mean". Groups smaller than this number have their means pulled towards the mean of the response across all cases. Default: 0
white_noise	(optional; numeric vector) Argument of the methods "mean", "rank", and "loo". Maximum white noise to add, expressed as a fraction of the range of the response variable. Range from 0 to 1. Default: 0.
seed	(optional; integer vector) Random seed to facilitate reproducibility when white_noise is not 0. If NULL, the function selects one at random, and the selected seed does not appear in the encoded variable names. Default: 0
overwrite	(optional; logical) If TRUE, the original predictors in df are overwritten with their encoded versions, but only one encoding method, smoothing, white noise, and seed are allowed. Otherwise, encoded predictors with their descriptive names are added to df. Default: FALSE
quiet	(optional; logical) If FALSE, messages generated during the execution of the function are printed to the console Default: FALSE

Value

data frame

Author(s)

Blas M. Benito, PhD

References

- Micci-Barreca, D. (2001) A Preprocessing Scheme for High-Cardinality Categorical Attributes in Classification and Prediction Problems. SIGKDD Explor. Newsl. 3, 1, 27-32. doi: 10.1145/507533.507538

See Also

Other target_encoding: target_encoding_mean()

Examples

```
data(
  vi,
  vi_predictors
  )
```

```
#subset to limit example run time
vi <- vi[1:1000, ]

#applying all methods for a continuous response
df <- target_encoding_lab(
  df = vi,
  response = "vi_numeric",
  predictors = "koppen_zone",
  methods = c(
    "mean",
    "loo",
    "rank"
  ),
  white_noise = c(0, 0.1, 0.2)
)

#identify encoded predictors
predictors.encoded <- grep(
  pattern = "*__encoded*",
  x = colnames(df),
  value = TRUE
)
```

---

target_encoding_mean *Target Encoding Methods*

Description

Target Encoding Methods

Usage

```
target_encoding_mean(
  df = NULL,
  response = NULL,
  predictor = NULL,
  encoded_name = NULL,
  smoothing = 0
)

target_encoding_rank(
  df = NULL,
  response = NULL,
  predictor = NULL,
  encoded_name = NULL,
  smoothing = 0
```

```
)

target_encoding_loo(
  df = NULL,
  response = NULL,
  predictor = NULL,
  encoded_name = NULL,
  smoothing = 0
)
```

Arguments

df	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
response	(optional, character string) Name of a numeric response variable in df. Default: NULL.
predictor	(required; string) Name of the categorical predictor to encode. Default: NULL
encoded_name	(required, string) Name of the encoded predictor. Default: NULL
smoothing	(optional; integer) Groups smaller than this number have their means pulled towards the mean of the response across all cases. Ignored by target_encoding_rank() and target_encoding_loo(). Default: 0

Value

data frame

See Also

Other target_encoding: target_encoding_lab()

Other target_encoding: target_encoding_lab()

Examples

```
data(vi)

#subset to limit example run time
vi <- vi[1:1000, ]

#mean encoding
#-------------

#without noise
df <- target_encoding_mean(
  df = vi,
  response = "vi_numeric",
  predictor = "soil_type",
  encoded_name = "soil_type_encoded"
)
```

```
plot(
  x = df$soil_type_encoded,
  y = df$vi_numeric,
  xlab = "encoded variable",
  ylab = "response"
)

#group rank
#----------

df <- target_encoding_rank(
  df = vi,
  response = "vi_numeric",
  predictor = "soil_type",
  encoded_name = "soil_type_encoded"
)

plot(
  x = df$soil_type_encoded,
  y = df$vi_numeric,
  xlab = "encoded variable",
  ylab = "response"
)


#leave-one-out
#-------------

#without noise
df <- target_encoding_loo(
  df = vi,
  response = "vi_numeric",
  predictor = "soil_type",
  encoded_name = "soil_type_encoded"
)

plot(
  x = df$soil_type_encoded,
  y = df$vi_numeric,
  xlab = "encoded variable",
  ylab = "response"
)
```

`toy` *One response and four predictors with varying levels of multicollinearity*

Description

Data frame with known relationship between responses and predictors useful to illustrate multicollinearity concepts. Created from vi using the code shown in the example.

Usage

```
data(toy)
```

Format

Data frame with 2000 rows and 5 columns.

Details

Columns:

- y: response variable generated from `a * 0.75 + b * 0.25 + noise`.
- a: most important predictor of y, uncorrelated with b.
- b: second most important predictor of y, uncorrelated with a.
- c: generated from `a + noise`.
- d: generated from `(a + b)/2 + noise`.

These are variance inflation factors of the predictors in `toy`. variable vif b 4.062 d 6.804 c 13.263 a 16.161

See Also

Other example_data: `vi`, `vi_predictors`, `vi_predictors_categorical`, `vi_predictors_numeric`

---

validate_data_cor *Validate Data for Correlation Analysis*

Description

Internal function to assess whether the input arguments `df` and `predictors` result in data dimensions suitable for pairwise correlation analysis.

If the number of rows in `df` is smaller than 10, an error is issued.

Usage

```
validate_data_cor(
  df = NULL,
  predictors = NULL,
  function_name = "collinear::validate_data_cor()",
  quiet = FALSE
)
```

Arguments

df	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
predictors	(optional; character vector) Names of the predictors to select from df. If omitted, all numeric columns in df are used instead. If argument response is not provided, non-numeric variables are ignored. Default: NULL
function_name	(optional, character string) Name of the function performing the check. Default: "collinear::validate_data_cor()"
quiet	(optional; logical) If FALSE, messages generated during the execution of the function are printed to the console Default: FALSE

Value

character vector: predictors names

See Also

Other data_validation: validate_data_vif(), validate_df(), validate_encoding_arguments(), validate_predictors(), validate_preference_order(), validate_response()

---

validate_data_vif *Validate Data for VIF Analysis*

Description

Internal function to assess whether the input arguments df and predictors result in data dimensions suitable for a VIF analysis.

If the number of rows in df is smaller than 10 times the length of predictors, the function either issues a message and restricts predictors to a manageable number, or returns an error.

Usage

```
validate_data_vif(
  df = NULL,
  predictors = NULL,
  function_name = "collinear::validate_data_vif()",
  quiet = FALSE
)
```

Arguments

df	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
predictors	(optional; character vector) Names of the predictors to select from df. If omitted, all numeric columns in df are used instead. If argument response is not provided, non-numeric variables are ignored. Default: NULL

function_name	(optional, character string) Name of the function performing the check. Default: "collinear::validate_data_vif()"
quiet	(optional; logical) If FALSE, messages generated during the execution of the function are printed to the console Default: FALSE

Value

character vector: predictors names

See Also

Other data_validation: validate_data_cor(), validate_df(), validate_encoding_arguments(), validate_predictors(), validate_preference_order(), validate_response()

---

validate_df *Validate Argument df*

Description

Internal function to validate the argument df and ensure it complies with the requirements of the package functions. It performs the following actions:

- Stops if 'df' is NULL.
- Stops if 'df' cannot be coerced to data frame.
- Stops if 'df' has zero rows.
- Removes geometry column if the input data frame is an "sf" object.
- Removes non-numeric columns with as many unique values as rows df has.
- Converts logical columns to numeric.
- Converts factor and ordered columns to character.
- Tags the data frame with the attribute validated = TRUE to let the package functions skip the data validation.

Usage

```
validate_df(df = NULL, quiet = FALSE)
```

Arguments

df	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
quiet	(optional; logical) If FALSE, messages generated during the execution of the function are printed to the console Default: FALSE

Value

data frame

See Also

Other data_validation: validate_data_cor(), validate_data_vif(), validate_encoding_arguments(), validate_predictors(), validate_preference_order(), validate_response()

Examples

```
data(vi)

#validating example data frame
vi <- validate_df(
  df = vi
)

#tagged as validated
attributes(vi)$validated
```

---

validate_encoding_arguments

Validates Arguments of target_encoding_lab()

Description

Internal function to validate configuration arguments for target_encoding_lab().

Usage

```
validate_encoding_arguments(
  df = NULL,
  response = NULL,
  predictors = NULL,
  methods = c("mean", "loo", "rank"),
  smoothing = 0,
  white_noise = 0,
  seed = 0,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

df	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
response	(optional, character string) Name of a numeric response variable in df. Default: NULL.
predictors	(optional; character vector) Names of the predictors to select from df. If omitted, all numeric columns in df are used instead. If argument response is not provided, non-numeric variables are ignored. Default: NULL

`methods`	(optional; character vector or NULL). Name of the target encoding methods. If NULL, target encoding is ignored, and `df` is returned with no modification. Default: c("loo", "mean", "rank")
`smoothing`	(optional; integer vector) Argument of the method "mean". Groups smaller than this number have their means pulled towards the mean of the response across all cases. Default: 0
`white_noise`	(optional; numeric vector) Argument of the methods "mean", "rank", and "loo". Maximum white noise to add, expressed as a fraction of the range of the response variable. Range from 0 to 1. Default: `0`.
`seed`	(optional; integer vector) Random seed to facilitate reproducibility when `white_noise` is not 0. If NULL, the function selects one at random, and the selected seed does not appear in the encoded variable names. Default: 0
`overwrite`	(optional; logical) If `TRUE`, the original predictors in `df` are overwritten with their encoded versions, but only one encoding method, smoothing, white noise, and seed are allowed. Otherwise, encoded predictors with their descriptive names are added to `df`. Default: FALSE
`quiet`	(optional; logical) If FALSE, messages generated during the execution of the function are printed to the console Default: FALSE

Value

list

See Also

Other data_validation: `validate_data_cor()`, `validate_data_vif()`, `validate_df()`, `validate_predictors()`, `validate_preference_order()`, `validate_response()`

Examples

```
validate_encoding_arguments(
  df = vi,
  response = "vi_numeric",
  predictors = vi_predictors
  )
```

`validate_predictors` *Validate Argument predictors*

Description

Internal function to validate the `predictors` argument. Requires the argument 'df' to be validated with `validate_df()`.

Validates the 'predictors' argument to ensure it complies with the requirements of the package functions. It performs the following actions:

- Stops if 'df' is NULL.
- Stops if 'df' is not validated.
- If 'predictors' is NULL, uses column names of 'df' as 'predictors' in the 'df' data frame.
- Print a message if there are names in 'predictors' not in the column names of 'df', and returns only the ones in 'df'.
- Stop if the number of numeric columns in 'predictors' is smaller than 'min_numerics'.
- Print a message if there are zero-variance columns in 'predictors' and returns a new 'predictors' argument without them.
- Tags the vector with the attribute `validated = TRUE` to let the package functions skip the data validation.

Usage

```
validate_predictors(
  df = NULL,
  response = NULL,
  predictors = NULL,
  quiet = FALSE
)
```

Arguments

`df`	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
`response`	(optional; character string or vector) Name/s of response variable/s in `df`. Used in target encoding when it names a numeric variable and there are categorical predictors, and to compute preference order. Default: NULL.
`predictors`	(optional; character vector) Names of the predictors to select from `df`. If omitted, all numeric columns in `df` are used instead. If argument `response` is not provided, non-numeric variables are ignored. Default: NULL
`quiet`	(optional; logical) If FALSE, messages generated during the execution of the function are printed to the console Default: FALSE

Value

character vector: predictor names

See Also

Other data_validation: `validate_data_cor()`, `validate_data_vif()`, `validate_df()`, `validate_encoding_arguments()`, `validate_preference_order()`, `validate_response()`

Examples

```
data(
  vi,
  vi_predictors
  )

#validating example data frame
vi <- validate_df(
  df = vi
)

#validating example predictors
vi_predictors <- validate_predictors(
  df = vi,
  predictors = vi_predictors
)

#tagged as validated
attributes(vi_predictors)$validated
```

---

validate_preference_order

Validate Argument preference_order

Description

Internal function to validate the argument `preference_order`.

Usage

```
validate_preference_order(
  predictors = NULL,
  preference_order = NULL,
  preference_order_auto = NULL,
  function_name = "collinear::validate_preference_order()",
  quiet = FALSE
)
```

Arguments

`predictors`	(optional; character vector) Names of the predictors to select from `df`. If omitted, all numeric columns in `df` are used instead. If argument `response` is not provided, non-numeric variables are ignored. Default: NULL
`preference_order`	(optional; string, character vector, output of `preference_order()`). Defines a priority order, from first to last, to preserve predictors during the selection process. Accepted inputs are:

- **"auto"** (default): if `response` is not NULL, calls `preference_order()` for internal computation.
- **character vector**: predictor names in a custom preference order.
- **data frame**: output of `preference_order()` from `response` of length one.
- **named list**: output of `preference_order()` from `response` of length two or more.
- **NULL**: disabled.

. Default: "auto"

`preference_order_auto`
: (required, character vector) names of the predictors in the automated preference order returned by `vif_select()` or `cor_select()`

`function_name`
: (optional, character string) Name of the function performing the check. Default: "collinear::validate_preference_order()"

`quiet`
: (optional; logical) If FALSE, messages generated during the execution of the function are printed to the console Default: FALSE

Value

character vector: ranked variable names

See Also

Other data_validation: `validate_data_cor()`, `validate_data_vif()`, `validate_df()`, `validate_encoding_arguments()`, `validate_predictors()`, `validate_response()`

Examples

```
data(
  vi,
  vi_predictors
  )

#validating example data frame
vi <- validate_df(
  df = vi
)

#validating example predictors
vi_predictors <- validate_predictors(
  df = vi,
  predictors = vi_predictors
)

#tagged as validated
attributes(vi_predictors)$validated

#validate preference order
my_order <- c(
  "swi_max",
```

```
  "swi_min",
  "swi_deviance" #wrong one
)

my_order <- validate_preference_order(
  predictors = vi_predictors,
  preference_order = my_order,
  preference_order_auto = vi_predictors
)

#has my_order first
#excludes wrong names
#all other variables ordered according to preference_order_auto
my_order
```

`validate_response` *Validate Argument response*

Description

Internal function to validate the argument `response`. Requires the argument 'df' to be validated with `validate_df()`.

Usage

```
validate_response(df = NULL, response = NULL, quiet = FALSE)
```

Arguments

`df`	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
`response`	(optional; character string or vector) Name/s of response variable/s in `df`. Used in target encoding when it names a numeric variable and there are categorical predictors, and to compute preference order. Default: NULL.
`quiet`	(optional; logical) If FALSE, messages generated during the execution of the function are printed to the console Default: FALSE

Value

character string: response name

See Also

Other data_validation: `validate_data_cor()`, `validate_data_vif()`, `validate_df()`, `validate_encoding_arguments`
`validate_predictors()`, `validate_preference_order()`

Examples

```
data(
  vi
)

#validating example data frame
vi <- validate_df(
  df = vi
)

#validating example predictors
response <- validate_response(
  df = vi,
  response = "vi_numeric"
)

#tagged as validated
attributes(response)$validated
```

---

vi *Example Data With Different Response and Predictor Types*

Description

The response variable is a Vegetation Index encoded in different ways to help highlight the package capabilities:

- `vi_numeric`: continuous vegetation index values in the range 0-1.
- `vi_counts`: simulated integer counts created by multiplying `vi_numeric` by 1000 and coercing the result to integer.
- `vi_binomial`: simulated binomial variable created by transforming `vi_numeric` to zeros and ones.
- `vi_categorical`: character variable with the categories "very_low", "low", "medium", "high", and "very_high", with thresholds located at the quantiles of `vi_numeric`.
- `vi_factor`: `vi_categorical` converted to factor.

The names of all predictors (continuous, integer, character, and factors) are in vi_predictors.

Usage

```
data(vi)
```

Format

Data frame with 30.000 rows and 68 columns.

See Also

vi_predictors

Other example_data: toy, vi_predictors, vi_predictors_categorical, vi_predictors_numeric

---

`vif_df` *Variance Inflation Factor*

Description

Computes the Variance Inflation Factor of numeric variables in a data frame.

This function computes the VIF (see section **Variance Inflation Factors** below) in two steps:

- Applies `base::solve()` to obtain the precision matrix, which is the inverse of the covariance matrix between all variables in `predictors`.
- Uses `base::diag()` to extract the diagonal of the precision matrix, which contains the variance of the prediction of each predictor from all other predictors, and represents the VIF.

Usage

```
vif_df(df = NULL, predictors = NULL, quiet = FALSE)
```

Arguments

`df`	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
`predictors`	(optional; character vector) Names of the predictors to select from `df`. If omitted, all numeric columns in `df` are used instead. If argument `response` is not provided, non-numeric variables are ignored. Default: NULL
`quiet`	(optional; logical) If FALSE, messages generated during the execution of the function are printed to the console Default: FALSE

Value

data frame; predictors names their VIFs

Variance Inflation Factors

The Variance Inflation Factor for a given variable a is computed as $1/(1 - R2)$, where $R2$ is the multiple R-squared of a multiple regression model fitted using a as response and all other predictors in the input data frame as predictors, as in $a = b + c + ...$

The square root of the VIF of a is the factor by which the confidence interval of the estimate for a in the linear model $y = a + b + c + ...$‘ is widened by multicollinearity in the model predictors.

The range of VIF values is (1, Inf]. The recommended thresholds for maximum VIF may vary depending on the source consulted, being the most common values, 2.5, 5, and 10.

References

- David A. Belsley, D.A., Kuh, E., Welsch, R.E. (1980). Regression Diagnostics: Identifying Influential Data and Sources of Collinearity. John Wiley & Sons. DOI: 10.1002/0471725153.

See Also

Other vif: `vif_select()`

Examples

```
data(
  vi,
  vi_predictors_numeric
)

#subset to limit run time
df <- vi[1:1000, ]

#apply pairwise correlation first
selection <- cor_select(
  df = df,
  predictors = vi_predictors_numeric,
  quiet = TRUE
)

#VIF data frame
df <- vif_df(
  df = df,
  predictors = selection
)

df
```

`vif_select` *Automated Multicollinearity Filtering with Variance Inflation Factors*

Description

This function automatizes multicollinearity filtering in data frames with numeric predictors by combining two methods:

- **Preference Order**: method to rank and preserve relevant variables during multicollinearity filtering. See argument `preference_order` and function `preference_order()`.
- **VIF-based filtering**: recursive algorithm to identify and remove predictors with a VIF above a given threshold.

When the argument `preference_order` is not provided, the predictors are ranked lower to higher VIF. The predictor selection resulting from this option, albeit diverse and uncorrelated, might not be the one with the highest overall predictive power when used in a model.

Please check the sections **Preference Order**, **Variance Inflation Factors**, and **VIF-based Filtering** at the end of this help file for further details.

Usage

```
vif_select(
  df = NULL,
  predictors = NULL,
  preference_order = NULL,
  max_vif = 5,
  quiet = FALSE
)
```

Arguments

`df`	(required; data frame, tibble, or sf) A data frame with responses and predictors. Default: NULL.
`predictors`	(optional; character vector) Names of the predictors to select from `df`. If omitted, all numeric columns in `df` are used instead. If argument `response` is not provided, non-numeric variables are ignored. Default: NULL
`preference_order`	(optional; string, character vector, output of `preference_order()`). Defines a priority order, from first to last, to preserve predictors during the selection process. Accepted inputs are: • **"auto"** (default): if `response` is not NULL, calls `preference_order()` for internal computation. • **character vector**: predictor names in a custom preference order. • **data frame**: output of `preference_order()` from `response` of length one. • **named list**: output of `preference_order()` from `response` of length two or more. • **NULL**: disabled. . Default: "auto"
`max_vif`	(optional, numeric) Maximum Variance Inflation Factor allowed during variable selection. Recommended values are between 2.5 and 10. Higher values return larger number of predictors with a higher multicollinearity. If NULL, the variance inflation analysis is disabled. Default: 5.
`quiet`	(optional; logical) If FALSE, messages generated during the execution of the function are printed to the console Default: FALSE

Value

- character vector if `response` is NULL or is a string.
- named list if `response` is a character vector.

Preference Order

This feature is designed to help protect important predictors during the multicollinearity filtering. It involves the arguments `preference_order` and `f`.

The argument `preference_order` accepts:

- : A character vector of predictor names in a custom order of preference, from first to last. This vector does not need to contain all predictor names, but only the ones relevant to the user.
- A data frame returned by `preference_order()`, which ranks predictors based on their association with a response variable.
- If NULL, and `response` is provided, then `preference_order()` is used internally to rank the predictors using the function `f`. If `f` is NULL as well, then `f_auto()` selects a proper function based on the data properties.

Variance Inflation Factors

The Variance Inflation Factor for a given variable a is computed as $1/(1 - R2)$, where $R2$ is the multiple R-squared of a multiple regression model fitted using a as response and all other predictors in the input data frame as predictors, as in $a = b + c + ...$

The square root of the VIF of a is the factor by which the confidence interval of the estimate for a in the linear model $y = a + b + c + ...$‘ is widened by multicollinearity in the model predictors.

The range of VIF values is (1, Inf]. The recommended thresholds for maximum VIF may vary depending on the source consulted, being the most common values, 2.5, 5, and 10.

VIF-based Filtering

The function `vif_select()` computes Variance Inflation Factors and removes variables iteratively, until all variables in the resulting selection have a VIF below `max_vif`.

If the argument `preference_order` is not provided, all variables are ranked from lower to higher VIF, as returned by `vif_df()`, and the variable with the higher VIF above `max_vif` is removed on each iteration.

If `preference_order` is defined, whenever two or more variables are above `max_vif`, the one higher in `preference_order` is preserved, and the next one with a higher VIF is removed. For example, for the predictors and preference order a and b, if any of their VIFs is higher than `max_vif`, then b will be removed no matter whether its VIF is lower or higher than a's VIF. If their VIF scores are lower than `max_vif`, then both are preserved.

Author(s)

Blas M. Benito, PhD

References

- David A. Belsley, D.A., Kuh, E., Welsch, R.E. (1980). Regression Diagnostics: Identifying Influential Data and Sources of Collinearity. John Wiley & Sons. DOI: 10.1002/0471725153.

See Also

Other vif: `vif_df()`

Examples

```
#subset to limit example run time
df <- vi[1:1000, ]
predictors <- vi_predictors[1:10]
predictors_numeric <- vi_predictors_numeric[1:10]

#predictors has mixed types
sapply(
  X = df[, predictors, drop = FALSE],
  FUN = class
)

#categorical predictors are ignored
x <- vif_select(
  df = df,
  predictors = predictors,
  max_vif = 2.5
)

x

#all these have a VIF lower than max_vif (2.5)
vif_df(
  df = df,
  predictors = x
)


#higher max_vif results in larger selection
x <- vif_select(
  df = df,
  predictors = predictors_numeric,
  max_vif = 10
)

x


#smaller max_vif results in smaller selection
x <- vif_select(
  df = df,
  predictors = predictors_numeric,
  max_vif = 2.5
)

x


#custom preference order
x <- vif_select(
  df = df,
  predictors = predictors_numeric,
```

```
  preference_order = c(
    "swi_mean",
    "soil_temperature_mean",
    "topo_elevation"
  ),
  max_vif = 2.5
)

x

#using automated preference order
df_preference <- preference_order(
  df = df,
  response = "vi_numeric",
  predictors = predictors_numeric
)

x <- vif_select(
  df = df,
  predictors = predictors_numeric,
  preference_order = df_preference,
  max_vif = 2.5
)

x


#categorical predictors are ignored
#the function returns NA
x <- vif_select(
  df = df,
  predictors = vi_predictors_categorical
)

x


#if predictors has length 1
#selection is skipped
#and data frame with one row is returned
x <- vif_select(
  df = df,
  predictors = predictors_numeric[1]
)

x
```

---

vi_predictors *All Predictor Names in Example Data Frame vi*

Description

All Predictor Names in Example Data Frame vi

Usage

```
data(vi_predictors)
```

Format

Character vector with predictor names.

See Also

vi

Other example_data: toy, vi, vi_predictors_categorical, vi_predictors_numeric

---

`vi_predictors_categorical` *All Categorical and Factor Predictor Names in Example Data Frame vi*

Description

All Categorical and Factor Predictor Names in Example Data Frame vi

Usage

```
data(vi_predictors_categorical)
```

Format

Character vector with predictor names.

See Also

vi

Other example_data: toy, vi, vi_predictors, vi_predictors_numeric

`vi_predictors_numeric` *All Numeric Predictor Names in Example Data Frame vi*

Description

All Numeric Predictor Names in Example Data Frame vi

Usage

```
data(vi_predictors_numeric)
```

Format

Character vector with predictor names.

See Also

vi

Other example_data: toy, vi, vi_predictors, vi_predictors_categorical

Index