

Package: igate (via r-universe)

June 7, 2026

Type Package

Title Guided Analytics for Testing Manufacturing Parameters

Version 0.3.3

Description An implementation of the initial guided analytics for parameter testing and controlband extraction framework. Functions are available for continuous and categorical target variables as well as for generating standardized reports of the conducted analysis. See <<https://github.com/stefan-stein/igate>> for more information on the technology.

URL <https://github.com/stefan-stein/igate>

BugReports <https://github.com/stefan-stein/igate/issues>

License GPL-3

Encoding UTF-8

LazyData true

SystemRequirements pandoc (>= 1.12.3) - <http://pandoc.org>

Depends R (>= 3.6.0),

Imports ggplot2, dplyr, grDevices, stringr, graphics, stats, knitr, xtable, kableExtra, rmarkdown

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Stefan Stein [aut, cre]

Maintainer Stefan Stein <s.stein@warwick.ac.uk>

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev make libharfbuzz-dev libicu-dev libpng-dev libuv1-dev libxml2-dev

Repository <https://cranhaven.r-universe.dev>

Date/Publication 2026-05-08 01:02:00 UTC

RemoteUrl <https://github.com/cranhaven/cranhaven.r-universe.dev>

RemoteRef package/igate

RemoteSha e8a6ce5ec1dd2e7802205860b0682f9a9aede402

RemoteSubdir igate

Contents

categorical.freqplot	2
categorical.igate	3
counting.test	5
igate	6
igate.regressions	8
report	9
resultsIris	11
robust.categorical.igate	12
validate	13
validatedObsIris	15
validationCountsIris	15
validationSummaryIris	16
Index	17

categorical.freqplot *Produces frequency plots (normed to density plots to account for different category sizes) for sanity check in categorical iGATE.*

Description

This function takes a data frame, a categorical target variable and a list of ssv and produces a density plot of each ssv and each category of the target variable. The output is written as .png file into the current working directory. Also, summary statistics are provided. The files can be saved into the current working directory. Consider changing the working directory to a new empty folder before running if you want to save a copy of the plots.

Usage

```
categorical.freqplot(df, target, ssv = NULL,
  outlier_removal_ssv = TRUE, savePlots = FALSE,
  image_directory = tempdir())
```

Arguments

df	Data frame to be analysed.
target	Categorical target variable to be analysed.
ssv	A vector of suspected sources of variation. These are the variables in df which we believe might have an influence on the target variable and will be tested. If no list of ssv is provided, the test will be performed on all numeric variables.

outlier_removal_ssv	Logical. Should outlier removal be performed for each ssv (default: TRUE)?
savePlots	Logical. If FALSE (the default) frequency plots will be output to the standard plotting device. If TRUE, frequency plots will be saved to image_directory as png files.
image_directory	Directory to which plots should be saved. This is only used if savePlots = TRUE and defaults to the temporary directory of the current R session, i.e. tempdir(). To save plots to the current working directory set savePlots = TRUE and image_directory = getwd().

Details

Frequency plots for each ssv against each category of the target are produced and saved to current working directory. Also a data frame with summary statistics is produced, see **Value** for details.

Value

The density plots of each category of target against each ssv are written as .png file into the current working directory. Also, a data frame with the following columns is output

Causes	The ssv that were analysed.
outliers_removed	How many outliers (with respect to this ssv) have been removed before drawing the plot?
observations_retained	After outlier removal was performed, how many observations were left and used to fit the model?
frequency_plot	Logical. Was plotting successful? No plot will be produced if a ssv is constant.

Examples

```
categorical.freqplot(mtcars, target = "cyl")
```

categorical.igate *igate function for categorical target variables*

Description

This function performs an initial Guided Analysis for parameter testing and controlband extraction (iGATE) for a categorical target variable on a dataset and returns those parameters found to be influential.

Usage

```
categorical.igate(df, versus = 8, target, best.cat, worst.cat,
  test = "w", ssv = NULL, outlier_removal_ssv = TRUE)
```

Arguments

<code>df</code>	Data frame to be analysed.
<code>versus</code>	How many Best of the Best and Worst of the Worst do we collect? By default, we will collect 8 of each.
<code>target</code>	Target variable to be analysed. Must be categorical. Use <code>igate</code> for continuous target.
<code>best.cat</code>	The best category. The versus BOB will be selected randomly from this category.
<code>worst.cat</code>	The worst category. The versus WOW will be selected randomly from this category.
<code>test</code>	Statistical hypothesis test to be used to determine influential process parameters. Choose between Wilcoxon Rank test ("w", default) and Student's t-test ("t").
<code>ssv</code>	A vector of suspected sources of variation. These are the variables in <code>df</code> which we believe might have an influence on the <code>target</code> variable and will be tested. If no list of <code>ssv</code> is provided, the test will be performed on all numeric variables.
<code>outlier_removal_ssv</code>	Logical. Should outlier removal be performed for each <code>ssv</code> (default: TRUE)?

Details

We collect the Best of the Best and the Worst of the Worst dynamically dependent on the current `ssv`. That means, for each `ssv` we first remove all the observations with missing values for that `ssv` from `df`. Then, based on the remaining observations, we randomly select `versus` observations from the the best category ("Best of the Best", short BOB) and `versus` observations from the worst category ("Worst of the Worst", short WOW). By default, we select 8 of each. Next, we compare BOB and WOW using the the counting method and the specified hypothesis test. If the distributions of the `ssv` in BOB and WOW are significantly different, the current `ssv` has been identified as influential to the `target` variable. An `ssv` is considered influential, if the test returns a count larger/ equal to 6 and/ or a p-value of less than 0.05. For the next `ssv` we again start with the entire dataset `df`, remove all the observations with missing values for that new `ssv` and then select our new BOB and WOW. In particular, for each `ssv` we might select different observations. This dynamic selection is necessary, because in case of an incomplete data set, if we select the same BOB and WOW for all the `ssv`, we might end up with many missing values for particular `ssv`. In that case the hypothesis test loses statistical power, because it is used on a smaller sample or worse, might fail altogether if the sample size gets too small.

For those `ssv` determined to be significant, control bands are extracted. The rationale is: If the value for an `ssv` is in the interval `[good_lower_bound,good_upper_bound]` the `target` is likely to be good. If it is in the interval `[bad_lower_bound,bad_upper_bound]`, the `target` is likely to be bad.

Furthermore some summary statistics are provided: `na_removed` tells us how many observations have been removed for a particular `ssv`. When selecting the versus BOB/ WOW, the selection is done randomly from within the best/ worst category, i.e. the versus BOB/ WOW are not uniquely determined. The randomness in the selection is quantified by `ties_best_cat`, `ties_worst_cat`, which gives the size of the best/ worst category respectively.

Value

A data frame with the following columns

Causes	Those ssv that have been found to be influential to the target variable.
Count	The value returned by the counting method.
p.value	The p-value of the hypothesis test performed, i.e. either of the Wilcoxon rank test (in case test = "w")
good_lower_bound	The lower bound for this Cause for good quality.
good_upper_bound	The upper bound for this Cause for good quality.
bad_lower_bound	The lower bound for this Cause for bad quality.
bad_upper_bound	The upper bound for this Cause for bad quality.
na_removed	How many missing values were in the data set for this Cause?
ties_best_cat	How many observations fall into the best category?
ties_worst_cat	How many observations fall into the worst category?

Examples

```
df <- mtcars
df$cyl <- as.factor(df$cyl)
categorical.igate(df, target = "cyl", best.cat = "8", worst.cat = "4")
```

counting.test	<i>Performs the counting test</i>
---------------	-----------------------------------

Description

This test is based on Tukey's "A Quick, Compact, Two-Sample Test to Duckworth's Specifications", Technometrics, Vol. 1, No. 1 (1959), p.31-48. The test is chosen here because of its easy interpretability.

Usage

```
counting.test(B, W)
```

Arguments

B, W Numeric vectors with best observations (B) and worst observations (W).

Details

We form `rbind(B,W)` and order it. If B and W differ significantly, ordering `rbind(B,W)` will find observations of one group at the top and observations of the other at the bottom. We then count how many observations of one group are at the top and how many of the other are at the bottom. The sum of the two values gives us the count test statistic. A critical value of `count >= 6` corresponds to a p-value of roughly 0.05 and is independent of sample size and distributional assumptions. These clustered observations at the top and bottom of the ordered list also determine the control bands `good_band_lower_bound`, `good_band_upper_bound`, `bad_band_lower_bound`,

bad_band_upper_bound: We look if observations from group B are at the top or bottom. The highest/ lowest values for observations of group B within that cluster are good_band_lower_bound and good_band_upper_bound. We proceed with group W respectively. If no such clusters form at the end of the ordered list, the control bands are set to -1.

Value

A data frame with the following columns

count	The count test statistic described in Tukey's paper, adjusted for tied observations. The original test
good_band_lower_bound	Lower bound for good observations (B).
good_band_upper_bound	Upper bound for good observations (B).
bad_band_lower_bound	Lower bound for bad observations (W).
bad_band_upper_bound	Upper bound for bad observations (W).

igate	<i>igate function for continuous target variables</i>
-------	---

Description

This function performs an initial Guided Analysis for parameter testing and controlband extraction (iGATE) on a dataset and returns those parameters found to be influential.

Usage

```
igate(df, versus = 8, target, test = "w", ssv = NULL,
      outlier_removal_target = TRUE, outlier_removal_ssv = TRUE,
      good_end = "low", savePlots = FALSE, image_directory = tempdir())
```

Arguments

df	Data frame to be analysed.
versus	How many Best of the Best and Worst of the Worst do we collect? By default, we will collect 8 of each.
target	Target variable to be analysed. Must be continuous. Use categorical.igate for categorical target.
test	Statistical hypothesis test to be used to determine influential process parameters. Choose between Wilcoxon Rank test ("w", default) and Student's t-test ("t").
ssv	A vector of suspected sources of variation. These are the variables in df which we believe might have an influence on the target variable and will be tested. If no list of ssv is provided, the test will be performed on all numeric variables.
outlier_removal_target	Logical. Should outliers (with respect to the target variable) be removed from df (default: TRUE)? Important: This only makes sense if no prior outlier removal has been performed on df, i.e. df still contains all the data. Otherwise calculation for outlier threshold will be falsified.

outlier_removal_ssv	Logical. Should outlier removal be performed for each ssv (default: TRUE)?
good_end	Are low (default) or high values of target variable good? This is needed to determine the control bands.
savePlots	Logical, only relevant if outlier_removal_target is TRUE. If savePlots == FALSE (the default) the boxplot of the target variable will be output to the standard output device for plots, usually the console. If TRUE, the boxplot will additionally be saved to image_directory as a png file.
image_directory	Directory to which plots should be saved. This is only used if savePlots = TRUE and defaults to the temporary directory of the current R session, i.e. tempdir(). To save plots to the current working directory set savePlots = TRUE and image_directory = getwd().

Details

We collect the Best of the Best and the Worst of the Worst dynamically dependent on the current ssv. That means, for each ssv we first remove all the observations with missing values for that ssv from df. Then, based on the remaining observations, we select versus observations with the best values for the target variable (“Best of the Best”, short BOB) and versus observations with the worst values for the target variable (“Worst of the Worst”, short WOW). By default, we select 8 of each. Next, we compare BOB and WOW using the the counting method and the specified hypothesis test. If the distributions of the ssv in BOB and WOW are significantly different, the current ssv has been identified as influential to the target variable. An ssv is considered influential, if the test returns a count larger/ equal to 6 and/ or a p-value of less than 0.05. For the next ssv we again start with the entire dataset df, remove all the observations with missing values for that new ssv and then select our new BOB and WOW. In particular, for each ssv we might select different observations. This dynamic selection is necessary, because in case of an incomplete data set, if we select the same BOB and WOW for all the ssv, we might end up with many missing values for particular ssv. In that case the hypothesis test loses statistical power, because it is used on a smaller sample or worse, might fail altogether if the sample size gets too small.

For those ssv determined to be significant, control bands are extracted. The rationale is: If the value for an ssv is in the interval [good_lower_bound,good_upper_bound] the target is likely to be good. If it is in the interval [bad_lower_bound,bad_upper_bound], the target is likely to be bad.

Furthermore some summary statistics are provided: When selecting the versus BOB/ WOW, tied values for target can mean that the versus BOB/ WOW are not uniquely determined. In that case we randomly select from the tied observations to give us exactly versus observations per group. ties_lower_end, cometition_lower_end, ties_upper_end, competition_upper_end quantify this randomness. How to interpret these values: *lower end* refers to the group whose target values are *low* and *upper end* to the one whose target values are high. For example if a low value for target is good, *lower end* refers to the BOB and *upper end* to the WOW. We determine the versus BOB/ WOW via

```
lower_end <- df[min_rank(df$target)<=versus,]
```

If there are tied observations, nrow(lower_end) can be larger than versus. In ties_lower_end we record how many observations in lower_end\$target have the *highest* value and in competition_lower_end we record for how many places they are competing, i.e. competing_for_lower <- versus -

(`nrow(lower_end) - ties_lower_end`). The values for `ties_upper_end` and `competition_upper_end` are determined analogously.

Value

A data frame with the following columns

Causes	Those ssv that have been found to be influential to the target variable.
Count	The value returned by the counting method.
p.value	The p-value of the hypothesis test performed, i.e. either of the Wilcoxon rank test (in case <code>test = "wilcoxon"</code>) or the t-test (in case <code>test = "t"</code>).
good_lower_bound	The lower bound for this Cause for good quality.
good_upper_bound	The upper bound for this Cause for good quality.
bad_lower_bound	The lower bound for this Cause for bad quality.
bad_upper_bound	The upper bound for this Cause for bad quality.
na_removed	How many missing values were in the data set for this Cause?
ties_lower_end	Number of tied observations at lower end of target when selecting the versus BOB/ WOW.
competition_lower_end	For how many positions are the <code>tied_obs_lower</code> competing?
ties_upper_end	Number of tied observations at upper end of target when selecting the versus BOB/ WOW.
competition_upper_end	For how many positions are the <code>tied_obs_upper</code> competing?
adjusted.p.values	The p.values adjusted via Bonferroni correction.

Examples

```
igate(iris, target = "Sepal.Length")
```

<code>igate.regressions</code>	<i>Produces the regression plots for sanity check in iGATE</i>
--------------------------------	--

Description

This function takes a data frame, a target variable and a list of ssv and produces a regression plot of each ssv against the target. The output can be written as .png file into the current working directory. Also, summary statistics are provided.

Usage

```
igate.regressions(df, target, ssv = NULL,
  outlier_removal_target = TRUE, outlier_removal_ssv = TRUE,
  savePlots = FALSE, image_directory = tempdir())
```

Arguments

<code>df</code>	Data frame to be analysed.
<code>target</code>	Target variable to be analysed.
<code>ssv</code>	A vector of suspected sources of variation. These are the variables in <code>df</code> which we believe might have an influence on the target variable and will be tested. If no list of ssv is provided, the test will be performed on all numeric variables.

outlier_removal_target	Logical. Should outliers (with respect to the target variable) be removed from df (default: TRUE)? Important: This only makes sense if no prior outlier removal has been performed on df, i.e. df still contains all the data. Otherwise calculation for outlier threshold will be falsified.
outlier_removal_ssv	Logical. Should outlier removal be performed for each ssv (default: TRUE)?
savePlots	Logical. If FALSE (the default) regression plots will be output to the standard plotting device. If TRUE, regression plots will additionally be saved to image_directory as png files.
image_directory	Directory to which plots should be saved. This is only used if savePlots = TRUE and defaults to the temporary directory of the current R session, i.e. tempdir(). To save plots to the current working directory set savePlots = TRUE and image_directory = getwd().

Details

Regression plots for each ssv against target are produced and saved to current working directory. Also a data frame with summary statistics is produced, see **Value** for details.

Value

The regression plots of target against each ssv are written as .png file into the current working directory. Also, a data frame with the following columns is output

Causes	The ssv that were analysed.
outliers_removed	How many outliers (with respect to this ssv) have been removed before fitting the linear model?
observations_retained	After outlier removal was performed, how many observations were left and used to fit the model?
regression_plot	Logical. Was fitting the model successful? It can fail, for example, if a ssv is constant.
r_squared	r ² value of model.
gradient, intercept	Gradient and intercept of fitted model.

Examples

```
igate.regressions(iris, target = "Sepal.Length")
```

report

Generates report about a conducted igate.

Description

Takes results from a previous igate and automatically generates a html report for it. Be aware that running this function will create an html document in your current working directory.

Usage

```
report(df, versus = 8, target, type = "continuous", test = "w",
       ssv = NULL, outlier_removal_target = TRUE,
       outlier_removal_ssv = TRUE, good_outcome = "low", results_path,
       validation = FALSE, validation_path = NULL,
       validation_counts = NULL, validation_summary = NULL,
       image_directory = tempdir(), output_name = NULL, output_directory)
```

Arguments

df	The data frame that was analysed with igate or categorical.igate .
versus	What value of versus was used?
target	What target was used?
type	Was igate (use type = "continuous") or categorical.igate (use type = "categorical") conducted?
test	Which hypothesis test was used alongside the counting method?
ssv	Which ssv have been used in the analysis? If NULL, it will be assumed that ssv = NULL was passed to igate or categorical.igate and all numeric variables in df will be used.
outlier_removal_target	Was outlier removal conducted for target? If type == "categorical" this is set to FALSE automatically.
outlier_removal_ssv	Was outlier removal conducted for each ssv?
good_outcome	Are "low" or "high" values of target good? Or, in case of a categorical target the name of the best category as a string.
results_path	Name of R object (as string) containing the results of igate or categorical.igate .
validation	Logical. Has validation of the results been performed?
validation_path	Name R object (as string) containing the validated observations, i.e. first data frame returned by validate .
validation_counts	Name of R object (as string) containing the counts from validation, i.e. the second data frame returned by validate .
validation_summary	Name of R object (as string) containing the summary of validation_path, i.e. the third data frame returned by validate .
image_directory	Directory which contains the plots from igate , igate.regressions etc.
output_name	Desired name of the output file. File extension .html will be added automatically if not supplied. If NULL will be *iGATE_Report.html*.
output_directory	Directory into which the report should be saved. To save to the current working directory, use <code>output_directory = getwd()</code> .

Value

An html file named "iGATE_Report.html" will be output to the current working directory, containing details about the conducted analysis. This includes a list of the analysed SSV, as well as tables with the results from `igate/ categorical.igate` and plots from `igate.regressions/ categorical.freqplot`.

Examples

```
## Example for categorical target variable
# If you want to conduct an igate analysis from scratch, running report
# is the last step and relies on executing the other functions in this package first.
# Run categorical.igate
df <- mtcars
df$cyl <- as.factor(df$cyl)
results <- categorical.igate(df, target = "cyl", best.cat = "8", worst.cat = "4")
# Produce density plots
# Suppose you only want to analyse further the first three identified ssv
results <- results[1:3,]
categorical.freqplot(mtcars, target = "cyl", ssv = results$Causes , savePlots = TRUE)

report(df = df, target = "cyl", type = "categorical", good_outcome = "8",
results_path = "results",
output_name = "testing_igate", output_directory = tempdir())
```

`resultsIris`*Example results data file to be used for example report generation.*

Description

This is the output of `resultsIris <- igate(iris, target = "Sepal.Length")`

Usage

```
resultsIris
```

Format

A data frame as described in the documentation of `igate`.

 robust.categorical.igate

Robust igate for categorical target variables

Description

This function performs a robust an initial Guided Analysis for parameter testing and controlband extraction (iGATE) for a categorical target variable by repeatedly running `categorical.igate` and only returning those parameters that are selected more often than a certain threshold.

Usage

```
robust.categorical.igate(df, versus = 8, target, best.cat, worst.cat,
  test = "w", ssv = NULL, outlier_removal_ssv = TRUE,
  iterations = 50, threshold = 0.5)
```

Arguments

<code>df</code>	Data frame to be analysed.
<code>versus</code>	How many Best of the Best and Worst of the Worst do we collect? By default, we will collect 8 of each.
<code>target</code>	Target variable to be analysed. Must be categorical. Use <code>igate</code> for continuous target.
<code>best.cat</code>	The best category. The versus BOB will be selected randomly from this category.
<code>worst.cat</code>	The worst category. The versus WOW will be selected randomly from this category.
<code>test</code>	Statistical hypothesis test to be used to determine influential process parameters. Choose between Wilcoxon Rank test ("w", default) and Student's t-test ("t").
<code>ssv</code>	A vector of suspected sources of variation. These are the variables in <code>df</code> which we believe might have an influence on the target variable and will be tested. If no list of <code>ssv</code> is provided, the test will be performed on all numeric variables.
<code>outlier_removal_ssv</code>	Logical. Should outlier removal be performed for each <code>ssv</code> (default: TRUE)?
<code>iterations</code>	Integer. How often should <code>categorical.igate</code> be performed? A message about how many iterations have been performed so far will be printed to the console every $0.1 \times \text{iterations}$ iterations.
<code>threshold</code>	Between 0 and 1. Only parameters that are selected at least $\text{floor}(\text{iterations} \times \text{threshold})$ times are returned.

Details

We collect the Best of the Best and the Worst of the Worst dynamically dependent on the current `ssv`. That means, for each `ssv` we first remove all the observations with missing values for that `ssv` from `df`. Then, based on the remaining observations, we randomly select versus observations from the the best category (“Best of the Best”, short BOB) and versus observations from the worst category (“Worst of the Worst”, short WOW). By default, we select 8 of each. Since this selection happens randomly, it is recommended to use `robust.categorical.igate` over `categorical.igate`. After the selection we compare BOB and WOW using the the counting method and the specified hypothesis test. If the distributions of the `ssv` in BOB and WOW are significantly different, the current `ssv` has been identified as influential to the target variable. An `ssv` is considered influential, if the test returns a count larger/ equal to 6 and/ or a p-value of less than 0.05. For the next `ssv` we again start with the entire dataset `df`, remove all the observations with missing values for that new `ssv` and then select our new BOB and WOW. In particular, for each `ssv` we might select different observations. This dynamic selection is necessary, because in case of an incomplete data set, if we select the same BOB and WOW for all the `ssv`, we might end up with many missing values for particular `ssv`. In that case the hypothesis test loses statistical power, because it is used on a smaller sample or worse, might fail altogether if the sample size gets too small.

For those `ssv` determined to be significant, control bands are extracted. The rationale is: If the value for an `ssv` is in the interval `[good_lower_bound,good_upper_bound]` the target is likely to be good. If it is in the interval `[bad_lower_bound,bad_upper_bound]`, the target is likely to be bad.

Value

A data frame with the summary statistics for those parameters that were selected at least `floor(iterations*threshold)` times:

Causes	Those <code>ssv</code> that have been found to be influential to the target variable.
<code>median_count</code>	The median value returned by the counting method for this parameter.
<code>median_p_value</code>	The median p-value of the hypothesis test performed, i.e. either of the Wilcoxon rank test (in c
<code>median_good_lower_bound</code>	The median lower bound for this Cause for good quality.
<code>median_good_upper_bound</code>	The median upper bound for this Cause for good quality.
<code>median_bad_lower_bound</code>	The median lower bound for this Cause for bad quality.
<code>median_bad_upper_bound</code>	The median upper bound for this Cause for bad quality.

Examples

```
robust.categorical.igate(mtcars, target = "cyl",
  best.cat = "8", worst.cat = "4", iterations = 50, threshold = 0.5)
```

 validate

Validates results after using `igate` or `categorical.igate`.

Description

Takes a new data frame to be used for validation and the causes and control bands obtained from `igate` or `categorical.igate` and returns all those observations that fall within these control bands.

Usage

```
validate(validation_df, target, causes, results_df, type = NULL)
```

Arguments

validation_df Data frame to be used for validation. It is recommended to use a different data frame from the one used in `igate/ categorical.igate`. The same data frame can be used if just a sanity check of the results is performed. This data frame must contain the target variable as well as all the causes determined by `igate/ categorical.igate`.

target Target variable that was used in `igate` or `categorical.igate`.

causes Causes determined by `igate` or `categorical.igate`. If you saved the results of `igate/ categorical.igate` in an object `results`, simply use `results$Causes` here.

results_df The data frame containing the results of `igate` or `categorical.igate`.

type The type of `igate` that was performed: either "continuous" or "categorical". If not provided function will try to guess the correct type based on the type of `validation_df[[target]]`.

Details

If a value of `Good_Count` or `Bad_count` is very low in the second data frame, it means that this cause is excluding a lot of observations from the first data frame. Consider re-running `validate` with this cause removed from `causes`.

Value

A list of three data frames is returned. The first data frame contains those observations in `validation_df` that fall into *all* the good resp. bad control bands specified in `results_df`. The columns are `target`, then one column for each of the causes and a new column `expected_quality` which is "good" if the observation falls into all the good control bands and "bad" if it falls into all the bad control bands.

The second data frame has three columns

Cause	Each of the causes.
Good_Count	If we selected all those observations that fall into the good band of this cause, how many observations would we have?
Bad_Count	If we selected all those observations that fall into the bad band of this cause, how many observations would we have?

The third data frame summarizes the first data frame: If `type = "continuous"` it has three columns:

<code>expected_quality</code>	Either "good" or "bad".
<code>max_target</code>	The maximum value for <code>target</code> for the observations with "good" expected quality resp. "bad" expected quality.
<code>min_target</code>	Minimum value of <code>target</code> for good resp. bad expected quality.

If `type = "categorical"` it has the following three columns:

expected_quality Either "good" or "bad".
 Category A list of categories of the observations with expected quality good resp. bad.
 Frequency A count how often the respective Category appears amongs the observations with good/ bad expected q

Examples

```
validate(iris, target = "Sepal.Length", causes = resultsIris$Causes, results_df = resultsIris)
```

validatedObsIris *validatedObsIris data set*

Description

Example validation data file to be used for example report generation.

Usage

```
validatedObsIris
```

Format

A data frame as described in the documentation of [validate](#).

Details

This is the output of

```
x <- validate(iris, target = "Sepal.Length", causes = resultsIris$Causes, results_df =
resultsIris)
validatedObsIris <- x[[1]]
```

validationCountsIris *validationCountsIris data set*

Description

Example validation data file to be used for example report generation.

Usage

```
validationCountsIris
```

Format

A data frame as described in the documentation of [validate](#).

Details

This is the output of

```
x <- validate(iris, target = "Sepal.Length", causes = resultsIris$Causes, results_df =
resultsIris)
validationCountsIris <- x[[2]]
```

validationSummaryIris *validationSummaryIris data set*

Description

Example validation data file to be used for example report generation.

Usage

```
validationSummaryIris
```

Format

A data frame as described in the documentation of [validate](#).

Details

This is the output of

```
x <- validate(iris, target = "Sepal.Length", causes = resultsIris$Causes, results_df =
resultsIris)
validationSummaryIris <- x[[3]]
```

Index

* datasets

- resultsIris, [11](#)
- validatedObsIris, [15](#)
- validationCountsIris, [15](#)
- validationSummaryIris, [16](#)

- categorical.freqplot, [2](#), [11](#)
- categorical.igate, [3](#), [6](#), [10–14](#)
- counting.test, [5](#)

- igate, [4](#), [6](#), [10–14](#)
- igate.regressions, [8](#), [11](#)

- report, [9](#)
- resultsIris, [11](#)
- robust.categorical.igate, [12](#)

- validate, [10](#), [13](#), [15](#), [16](#)
- validatedObsIris, [15](#)
- validationCountsIris, [15](#)
- validationSummaryIris, [16](#)