# Package ‘insurancerating’

March 31, 2020

<table>
<thead>
<tr>
<th>Type</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Title</strong></td>
<td>Analytic Insurance Rating Techniques</td>
</tr>
<tr>
<td><strong>Version</strong></td>
<td>0.5.2</td>
</tr>
<tr>
<td><strong>VignetteBuilder</strong></td>
<td>knitr</td>
</tr>
<tr>
<td><strong>Maintainer</strong></td>
<td>Martin Haringa <a href="mailto:mtharinga@gmail.com">mtharinga@gmail.com</a></td>
</tr>
</tbody>
</table>

**Description**  Methods for insurance rating. It provides a data driven strategy for the construction of insurance tariff classes. This strategy is based on the work by Antonio and Valdez (2012) <doi:10.1007/s10182-011-0152-7>. The package also adds functionality showing additional lines for the reference categories in the levels of the coefficients in the output of a generalized linear regression analysis. In addition it implements a procedure determining the level of a factor with the largest exposure, and thereafter changing the base level of the factor to this level.

**License**  GPL (>= 2)

**URL**  [https://github.com/mharinga/insurancerating](https://github.com/mharinga/insurancerating), [https://mharinga.github.io/insurancerating/](https://mharinga.github.io/insurancerating/)

**Encoding**  UTF-8

**LazyData**  true

**RoxygenNote**  7.1.0

**Imports**  classInt, data.table, dplyr, evtree, ggplot2, lubridate, mgcv, patchwork, stringr

**Depends**  R (>= 3.3)

**Suggests**  knitr, rmarkdown, scales, testthat

**NeedsCompilation**  no

**Author**  Martin Haringa [aut, cre]

**Repository**  CRAN

**Date/Publication**  2020-03-30 22:30:02 UTC
Description

Takes an object produced by `construct_tariff_classes()`, and plots the fitted GAM. In addition the constructed tariff classes are shown.

Usage

```r
## S3 method for class 'constructtariffclasses'
autoplot(
  x,
  conf_int = FALSE,
  color_gam = "steelblue",
  show_observations = FALSE,
) autoplot.constructtariffclasses
```

Automatically create a `ggplot` for objects obtained from `construct_tariff_classes()`
autoplot.constructtariffclasses

    color_splits = "grey50",
    size_points = 1,
    color_points = "black",
    rotate_labels = FALSE,
    remove_outliers = NULL

Arguments

  x              constructtariffclasses object produced by construct_tariff_classes
  conf_int       determines whether 95% confidence intervals will be plotted. The default is conf_int = FALSE
  color_gam      a color can be specified either by name (e.g.: "red") or by hexadecimal code (e.g.: "#FF1234") (default is "steelblue")
  show_observations add observed frequency/severity points for each level of the variable for which tariff classes are constructed
  color_splits   change the color of the splits in the graph ("grey50" is default)
  size_points    size for points (1 is default)
  color_points   change the color of the points in the graph ("black" is default)
  rotate_labels  rotate x-labels 45 degrees (this might be helpful for overlapping x-labels)
  remove_outliers do not show observations above this number in the plot. This might be helpful for outliers.

Value

  a ggplot object

Author(s)

  Martin Haringa

Examples

  ## Not run:
  library(ggplot2)
  library(dplyr)
  fit_gam(MTPL, nclaims = nclaims, x = age_policyholder, exposure = exposure) %>%
          construct_tariff_classes(.) %>%
          autoplot(. , show_observations = TRUE)

  ## End(Not run)
autoplot.fitgam

Automatically create a ggplot for objects obtained from fit_gam()

Description

Takes an object produced by fit_gam(), and plots the fitted GAM.

Usage

```r
## S3 method for class 'fitgam'
autoplot(
x, conf_int = FALSE, color_gam = "steelblue", show_observations = FALSE, x_stepsize = NULL, size_points = 1, color_points = "black", rotate_labels = FALSE, remove_outliers = NULL)
```

Arguments

- `x`: fitgam object produced by fit_gam
- `conf_int`: determines whether 95% confidence intervals will be plotted. The default is `conf_int = FALSE`
- `color_gam`: a color can be specified either by name (e.g.: "red") or by hexadecimal code (e.g.: "#FF1234") (default is "steelblue")
- `show_observations`: add observed frequency/severity points for each level of the variable for which tariff classes are constructed
- `x_stepsize`: set step size for labels horizontal axis
- `size_points`: size for points (1 is default)
- `color_points`: change the color of the points in the graph ("black" is default)
- `rotate_labels`: rotate x-labels 45 degrees (this might be helpful for overlapping x-labels)
- `remove_outliers`: do not show observations above this number in the plot. This might be helpful for outliers.

Value

a ggplot object
Author(s)

Martin Haringa

Examples

```r
## Not run:
library(ggplot2)
library(dplyr)
fit_gam(MTPL, nclaims = nclaims, x = age_policyholder, exposure = exposure) %>%
  autoplot(. , show_observations = TRUE)
## End(Not run)
```

autoclot.univ_all  

Automatically create a ggplot for objects obtained from `univariate_all()`

Description

Takes an object produced by `univariate_all()`, and plots the available input.

Usage

```r
## S3 method for class 'univ_all'
autoplot(
x,
show_plots = c(1, 2, 3, 4),
ncol = 1,
background = TRUE,
labels = TRUE,
sort = FALSE,
dec.mark = ",",
color = "dodgerblue",
color_bg = "#E7B800"
)
```

Arguments

- `x`  
  - `univ_all` object produced by `univariate_all()`
- `show_plots`  
  - numeric vector of plots to be shown (default is c(1,2,3,4)), there are six available plots:
    - 1. frequency (i.e. number of claims / exposure)
    - 2. average severity (i.e. severity / number of claims)
    - 3. risk premium (i.e. severity / exposure)
    - 4. loss ratio (i.e. severity / premium)
• 5. average premium (i.e. premium / exposure)
• 6. exposure

ncol: number of columns in output (default is 1)
background: show exposure as a background histogram (default is TRUE)
labels: show labels with the exposure (default is TRUE)
sort: sort (or order) risk factor into descending order by exposure (default is FALSE)
dec.mark: control the format of the decimal point, as well as the mark between intervals before the decimal point, choose either "," (default) or "."
color: change the color of the points and line ("dodgerblue" is default)
color_bg: change the color of the histogram ("#E7B800" is default)

Value
a ggplot2 object

Examples

library(ggplot2)
x <- univariate_all(MTPL2, x = area, severity = amount, nclaims = nclaims, exposure = exposure)
autoplot(x)
autoplot(x, show_plots = c(6,1), background = FALSE, sort = TRUE)

## S3 method for class 'univ_avgpremium'
autoplot(
x,
    background = TRUE,
    labels = TRUE,
    sort = FALSE,
    dec.mark = ",",
    color = "dodgerblue",
    color_bg = "#E7B800"
)

Description
Takes an object produced by univariate_average_premium(), and plots the average premium.

Usage

autoplot.univ_avgpremium

Automatically create a ggplot for objects obtained from univariate_average_premium()
Arguments

- **x**: `univ_avgpremium` object produced by `univariate_average_premium()`
- **background**: show exposure as a histogram (default is TRUE)
- **labels**: show labels with the exposure (default is TRUE)
- **sort**: sort (or order) risk factor into descending order by exposure (default is FALSE)
- **dec.mark**: control the format of the decimal point, as well as the mark between intervals before the decimal point, choose either "," (default) or "."
- **color**: change the color of the points and line ("dodgerblue" is default)
- **color_bg**: change the color of the histogram ("#E7B800" is default)

Value

a `ggplot2` object

Examples

```r
library(ggplot2)
x <- univariate_average_premium(MTPL2, x = area, premium = premium, exposure = exposure)
autoplot(x)
```

Description

Takes an object produced by `univariate_average_severity()`, and plots the claim frequency.

Usage

```r
# S3 method for class 'univ_avgsegv'
autoplot(
x,
background = TRUE,
labels = TRUE,
sort = FALSE,
dec.mark = ",",
color = "dodgerblue",
color_bg = "#E7B800"
)
```
autoplot.univ_exposure

Arguments

- **x**: univ_avgsev object produced by univariate_average_severity()
- **background**: show exposure as a histogram (default is TRUE)
- **labels**: show labels with the exposure (default is TRUE)
- **sort**: sort (or order) risk factor into descending order by exposure (default is FALSE)
- **dec.mark**: control the format of the decimal point, as well as the mark between intervals before the decimal point, choose either "," (default) or "."
- **color**: change the color of the points and line ("dodgerblue" is default)
- **color_bg**: change the color of the histogram ("#E7B800" is default)

Value

a ggplot2 object

Examples

```r
library(ggplot2)
x <- univariate_average_severity(MTPL2, x = area, severity = amount,
                                 nclaims = nclaims, premium = premium)
autoplot(x)
```

Description

Takes an object produced by univariate_exposure(), and plots the exposure.

Usage

```r
# S3 method for class 'univ_exposure'
autoplot(
  x,
  labels = TRUE,
  sort = FALSE,
  coord_flip = TRUE,
  dec.mark = ",",
  color_bg = "#E7B800"
)
```
Arguments

- **x**: univ_exposure object produced by `univariate_exposure()`
- **labels**: show labels with the exposure (default is TRUE)
- **sort**: sort (or order) risk factor into descending order by exposure (default is FALSE)
- **coord_flip**: flip cartesian coordinates so that horizontal becomes vertical, and vertical, horizontal (default is TRUE)
- **dec.mark**: control the format of the decimal point, as well as the mark between intervals before the decimal point, choose either "," (default) or "."
- **color_bg**: change the color of the histogram ("#E7B800" is default)

Value

- a ggplot2 object

Examples

```r
library(ggplot2)
x <- univariate_exposure(MTPL2, area, exposure)
autoplot(x, sort = TRUE)
autoplot(x, coord_flip = FALSE)
```

Description

Takes an object produced by `univariate_frequency()`, and plots the claim frequency.

Usage

```r
# S3 method for class 'univ_freq'
autoplot(
  x,
  background = TRUE,
  labels = TRUE,
  sort = FALSE,
  dec.mark = ",",
  color = "dodgerblue",
  color_bg = "#E7B800"
)
```
Arguments

- **x**: univ_freq object produced by univariate_frequency()
- **background**: show exposure as a histogram (default is TRUE)
- **labels**: show labels with the exposure (default is TRUE)
- **sort**: sort (or order) risk factor into descending order by exposure (default is FALSE)
- **dec.mark**: control the format of the decimal point, as well as the mark between intervals before the decimal point, choose either "," (default) or "."
- **color**: change the color of the points and line ("dodgerblue" is default)
- **color_bg**: change the color of the histogram ("#E7B800" is default)

Value

- a ggplot2 object

Examples

```r
library(ggplot2)
x <- univariate_frequency(MTPL2, x = area, nclaims = nclaims, exposure = exposure)
autoplot(x)
autoplot(x, sort = TRUE, dec.mark = ",", color_bg = "mediumseagreen")
```

Description

Automatically create a ggplot for objects obtained from univariate_loss_ratio().

Usage

```r
# S3 method for class 'univ_lossratio'
autoplot(
  x,
  background = TRUE,
  labels = TRUE,
  sort = FALSE,
  dec.mark = ",",
  color = "dodgerblue",
  color_bg = "#E7B800"
)
```
### Arguments

- **x**: `univ_lossratio` object produced by `univariate_loss_ratio()`
- **background**: show exposure as a histogram (default is `TRUE`)
- **labels**: show labels with the exposure (default is `TRUE`)
- **sort**: sort (or order) risk factor into descending order by exposure (default is `FALSE`)
- **dec.mark**: control the format of the decimal point, as well as the mark between intervals before the decimal point, choose either "," (default) or "."
- **color**: change the color of the points and line ("dodgerblue" is default)
- **color_bg**: change the color of the histogram ("#E7B800" is default)

### Value

A `ggplot2` object

### Examples

```r
library(ggplot2)
x <- univariate_loss_ratio(MTPL2, x = area, severity = amount, premium = premium)
apputplot(x)
```

---

### Description

Takes an object produced by `univariate_risk_premium()`, and plots the average risk premium.

### Usage

```r
## S3 method for class 'univ_premium'
apputplot(
x,
background = TRUE,
labels = TRUE,
sort = FALSE,
dec.mark = ",",
color = "dodgerblue",
color_bg = "#E7B800"
)
```
Arguments

- x: univ_premium object produced by univariate_risk_premium()
- background: show exposure as a histogram (default is TRUE)
- labels: show labels with the exposure (default is TRUE)
- sort: sort (or order) risk factor into descending order by exposure (default is FALSE)
- dec.mark: control the format of the decimal point, as well as the mark between intervals before the decimal point, choose either "," (default) or "."
- color: change the color of the points and line ("dodgerblue" is default)
- color_bg: change the color of the histogram ("#E7B800" is default)

Value

A ggplot2 object

Examples

library(ggplot2)
x <- univariate_risk_premium(MTPL2, x = area, severity = amount, exposure = exposure)
autoplot(x)

Description

This function specifies the first level of a factor to the level with the largest exposure. Levels of factors are sorted using an alphabetic ordering. If the factor is used in a regression context, then the first level will be the reference. For insurance applications it is common to specify the reference level to the level with the largest exposure.

Usage

biggest_reference(x, weight)

Arguments

- x: an unordered factor
- weight: a vector containing weights (e.g. exposure). Should be numeric.

Value

A factor of the same length as x
**construct_tariff_classes**

**Author(s)**

Martin Haringa

**References**


**Examples**

```r
## Not run:
library(dplyr)
df <- chickwts %>%
  mutate_if(is.character, as.factor) %>%
  mutate_if(is.factor, list(~biggest_reference(., weight)))

## End(Not run)
```

**construct_tariff_classes**

*Construct insurance tariff classes*

**Description**

Constructs insurance tariff classes to fitgam objects produced by fit_gam. The goal is to bin the continuous risk factors such that categorical risk factors result which capture the effect of the covariate on the response in an accurate way, while being easy to use in a generalized linear model (GLM).

**Usage**

```r
construct_tariff_classes(
  object,
  alpha = 0,
  niterations = 10000,
  ntrees = 200,
  seed = 1
)
```

**Arguments**

- **object**: fitgam object produced by fit_gam
- **alpha**: complexity parameter. The complexity parameter (alpha) is used to control the number of tariff classes. Higher values for alpha render less tariff classes. (alpha = 0 is default).
- **niterations**: in case the run does not converge, it terminates after a specified number of iterations defined by niterations.
ntrees the number of trees in the population.
seed an numeric seed to initialize the random number generator (for reproducibility).

Details
Evolutionary trees are used as a technique to bin the fitgam object produced by fit_gam into risk homogeneous categories. This method is based on the work by Henckaerts et al. (2018). See Grubinger et al. (2014) for more details on the various parameters that control aspects of the evtree fit.

Value
A list with components
- prediction data frame with predicted values
- x name of continuous risk factor for which tariff classes are constructed
- model either 'frequency', 'severity' or 'burning'
- data data frame with predicted values and observed values
- x_obs observations for continuous risk factor
- splits vector with boundaries of the constructed tariff classes
- tariff_classes values in vector x coded according to which constructed tariff class they fall

Author(s)
Martin Haringa

References

Examples
```r
## Not run:
library(dplyr)
fit_gam(MTPL, nclaims = nclaims, x = age_policyholder, exposure = exposure) %>%
  construct_tariff_classes(.)

## End(Not run)
```
fisher

Fisher's natural breaks classification

Description

The function provides an interface to finding class intervals for continuous numerical variables, for example for choosing colours for plotting maps.

Usage

fisher(vec, n = 7, diglab = 2)

Arguments

.vec a continuous numerical variable
.n number of classes required (n = 7 is default)
.diglab number of digits (n = 2 is default)

Details

The "fisher" style uses the algorithm proposed by W. D. Fisher (1958) and discussed by Slocum et al. (2005) as the Fisher-Jenks algorithm. This function is adopted from the classInt package.

Value

Vector with clustering

Author(s)

Martin Haringa

References


**Description**

Fits a generalized additive model (GAM) to continuous risk factors in one of the following three types of models: the number of reported claims (claim frequency), the severity of reported claims (claim severity) or the burning cost (i.e. risk premium or pure premium).

**Usage**

```r
default). See details section.

model = "frequency",
round_x = NULL
)

Arguments

data: data.frame of an insurance portfolio
nclaims: column in data with number of claims
x: column in data with continuous risk factor
exposure: column in data with exposure
amount: column in data with claim amount
pure_premium: column in data with pure premium
model: choose either 'frequency', 'severity' or 'burning' (model = 'frequency' is default). See details section.
round_x: round elements in column x to multiple of round_x. This gives a speed enhancement for data containing many levels for x.

Details

The 'frequency' specification uses a Poisson GAM for fitting the number of claims. The logarithm of the exposure is included as an offset, such that the expected number of claims is proportional to the exposure.

The 'severity' specification uses a lognormal GAM for fitting the average cost of a claim. The average cost of a claim is defined as the ratio of the claim amount and the number of claims. The number of claims is included as a weight.

The 'burning' specification uses a lognormal GAM for fitting the pure premium of a claim. The pure premium is obtained by multiplying the estimated frequency and the estimated severity of claims. The word burning cost is used here as equivalent of risk premium and pure premium.
Value

A list with components

- **prediction**: data frame with predicted values
- **x**: name of continuous risk factor
- **model**: either 'frequency', 'severity' or 'burning'
- **data**: data frame with predicted values and observed values
- **x_obs**: observations for continuous risk factor

Author(s)

Martin Haringa

References


Examples

```r
fit_gam(MTPL, nclaims = nclaims, x = age_policyholder, exposure = exposure)
```

---

**get_splits**

Get splits from partykit object

Description

Get splits from partykit object

Usage

```r
get_splits(x)
```

Arguments

- **x**: A party object.
**Description**

A dataset containing the age, number of claims, and exposure of almost 33,000 policyholders

**Usage**

MTPL

**Format**

A data frame with 32,731 rows and 4 variables:

- **age_policyholder** age of policyholder, in years.
- **nclaims** number of claims.
- **exposure** exposure, for example, if a vehicle is insured as of July 1 for a certain year, then during that year, this would represent an exposure of 0.5 to the insurance company.
- **amount** claim amount in Euros.

**Author(s)**

Martin Haringa

**Source**

The data is derived from the portfolio of a large Dutch motor insurance company.
Format

A data frame with 3,000 rows and 6 variables:

- **customer_id**: customer id
- **area**: region where customer lives
- **nclaims**: number of claims
- **amount**: claim amount (severity)
- **exposure**: exposure
- **premium**: earned premium

Author(s)

Martin Haringa

Source

The data is derived from the portfolio of a large Dutch motor insurance company.

---

**period_to_months**  
*Split period to months*

Description

The function splits rows with a time period longer than one month to multiple rows with a time period of exactly one month each. Values in numeric columns (e.g. exposure or premium) are divided over the months proportionately.

Usage

```r
period_to_months(df, begin, end, ...)
```

Arguments

- **df**: data.frame
- **begin**: column in df with begin dates
- **end**: column in df with end dates
- **...**: numeric columns in df to split

Details

In insurance portfolios it is common that rows relate to periods longer than one month. This is for example problematic in case exposures per month are desired.

Since insurance premiums are constant over the months, and do not depend on the number of days per month, the function assumes that each month has the same number of days (i.e. 30).
rating_factors

Value
data.frame with same columns as in df, and one extra column called id

Author(s)
Martin Haringa

Examples
library(lubridate)
portfolio <- data.frame(
  begin1 = ymd(c("2014-01-01", "2014-01-01")),
  end = ymd(c("2014-03-14", "2014-05-10")),
  termination = ymd(c("2014-03-14", "2014-05-10")),
  exposure = c(0.2025, 0.3583),
  premium = c(125, 150))
period_to_months(portfolio, begin1, end, premium, exposure)

rating_factors Include reference group in regression output

Description
This extracts coefficients in terms of the original levels of the coefficients rather than the coded variables.

Usage
rating_factors(model, colname = "estimate", exponentiate = TRUE)

Arguments
model a (generalized) linear model fit
colname name of column with estimates. Defaults to "estimate".
exponentiate Logical indicating whether or not to exponentiate the the coefficient estimates. Defaults to TRUE.

Details
This function is adopted from the dummy.coefstats function. Our adoption prints a data.frame as output. Categorical variables should be changed to factors in the data.frame used to fit the (generalized) linear model.

Value
data.frame


**univariate_all**

**Author(s)**

Martin Haringa

**Examples**

```r
g1 <- glm(nclaims ~ age_policyholder, family = "poisson", data = MTPL)
rating_factors(g1)
```

**Description**

Univariate analysis for discrete risk factors in an insurance portfolio. The following summary statistics are calculated:

- frequency (i.e. number of claims / exposure)
- average severity (i.e. severity / number of claims)
- risk premium (i.e. severity / exposure)
- loss ratio (i.e. severity / premium)
- average premium (i.e. premium / exposure)

If input arguments are not specified, the summary statistics related to these arguments are ignored.

**Usage**

```r
univariate_all(
  df,
  x,
  severity = NULL,
  nclaims = NULL,
  exposure = NULL,
  premium = NULL
)
```

**Arguments**

- `df` data.frame with insurance portfolio
- `x` column in df with risk factor
- `severity` column in df with severity (default is NULL)
- `nclaims` column in df with number of claims (default is NULL)
- `exposure` column in df with exposure (default is NULL)
- `premium` column in df with premium (default is NULL)
Value
An list of class `univ_all` with components
- `df`: data frame
- `xvar`: name of column in df with risk factor
- `severity`: name of column in df with severity
- `nclaims`: name of column in df with number of claims
- `exposure`: name of column in df with exposure
- `premium`: name of column in df with premium

Examples
```r
univariate_all(MTPL2, x = area, severity = amount, nclaims = nclaims, 
exposure = exposure, premium = premium)
# The summary statistics related to premium are not calculated
univariate_all(MTPL2, x = area, severity = amount, nclaims = nclaims, exposure = exposure)
```

Description
Univariate average premium

Average premium for discrete risk factors in an insurance portfolio. For each level of the risk factor the average premium is equal to the ratio between the premium and the exposure.

Usage
```r
univariate_average_premium(
  df, 
  x, 
  premium, 
  exposure, 
  nclaims = NULL, 
  severity = NULL 
)
```

Arguments
- `df`: data.frame with insurance portfolio
- `x`: column in df with risk factor
- `premium`: column in df with premium (default is NULL)
- `exposure`: column in df with exposure
- `nclaims`: column in df with number of claims
- `severity`: column in df with severity (default is NULL)
**univariate_averageSeverity**

**Value**

An list of class `univ_lossratio` with components

- `df`: data frame with average premium
- `xvar`: name of column in df with risk factor
- `severity`: name of column in df with severity
- `nclaims`: name of column in df with number of claims
- `exposure`: name of column in df with exposure
- `premium`: name of column in df with premium

**Author(s)**

Martin Haringa

**Examples**

```
univariate_average_premium(MTPL2, x = area, premium = premium, exposure = exposure)
```

**Description**

Univariate average claim severity

Average claim frequency for discrete risk factors in an insurance portfolio. For each level of the risk factor the average claim severity is equal to the ratio between the severity and the number of claims.

**Usage**

```
univariate_average_severity(
  df,
  x,
  severity,
  nclaims,
  exposure = NULL,
  premium = NULL
)
```

**Arguments**

- `df`: data.frame with insurance portfolio
- `x`: column in df with risk factor
- `severity`: column in df with severity (default is NULL)
- `nclaims`: column in df with number of claims
- `exposure`: column in df with exposure
- `premium`: column in df with premium (default is NULL)
univariate_exposure

Value
An list of class `univ_freq` with components

- **df**: data frame with claim frequency
- **xvar**: name of column in df with risk factor
- **severity**: name of column in df with severity
- **nclaims**: name of column in df with number of claims
- **exposure**: name of column in df with exposure
- **premium**: name of column in df with premium

Author(s)
Martin Haringa

Examples

```r
univariate_average_severity(MTPL2, x = area, severity = amount, nclaims = nclaims, premium = premium)
```

---

### Description

Exposure for discrete risk factors in an insurance portfolio.

### Usage

```r
univariate_exposure(df, x, exposure = exposure)
```

### Arguments

- **df**: data.frame with insurance portfolio
- **x**: column in df with risk factor
- **exposure**: column in df with exposure

### Value

An list of class `univ_lossratio` with components

- **df**: data frame with average premium
- **xvar**: name of column in df with risk factor
- **exposure**: name of column in df with exposure
univariate_frequency

Author(s)
Martin Haringa

Examples
univariate_exposure(MTPL2, area, exposure)

univariate_frequency  Univariate claim frequency

Description
Claim frequency for discrete risk factors in an insurance portfolio. For each level of the risk factor
the claim frequency is equal to the ratio between the number of claims and the exposure.

Usage
univariate_frequency(df, x, nclaims, exposure, severity = NULL, premium = NULL)

Arguments
df  data.frame with insurance portfolio
x   column in df with risk factor
nclaims column in df with number of claims
exposure column in df with exposure
severity column in df with severity (default is NULL)
premium column in df with premium (default is NULL)

Value
An list of class univ_freq with components
df  data frame with claim frequency
xvar  name of column in df with risk factor
nclaims name of column in df with number of claims
exposure name of column in df with exposure
severity name of column in df with severity
premium name of column in df with premium

Author(s)
Martin Haringa

Examples
univariate_frequency(MTPL2, x = area, nclaims = nclaims, exposure = exposure)
univariate_loss_ratio  Univariate loss ratio

Description

Average loss ratio for discrete risk factors in an insurance portfolio. For each level of the risk factor the loss ratio is equal to the ratio between the severity and the earned premium.

Usage

univariate_loss_ratio(
  df,
  x,
  severity,
  premium,
  exposure = NULL,
  nclaims = NULL
)

Arguments

- df: data.frame with insurance portfolio
- x: column in df with risk factor
- severity: column in df with severity (default is NULL)
- premium: column in df with premium (default is NULL)
- exposure: column in df with exposure
- nclaims: column in df with number of claims

Value

An list of class univ_lossratio with components

- df: data frame with average loss ratio
- xvar: name of column in df with risk factor
- severity: name of column in df with severity
- nclaims: name of column in df with number of claims
- exposure: name of column in df with exposure
- premium: name of column in df with premium

Author(s)

Martin Haringa

Examples

univariate_loss_ratio(MTPL2, x = area, severity = amount, premium = premium)
univariate_risk_premium

Univariate risk premium

Description
Risk premium for discrete risk factors in an insurance portfolio. For each level of the risk factor the risk premium is equal to the ratio between the severity and the exposure.

Usage
univariate_risk_premium(
  df, 
  x, 
  severity, 
  exposure, 
  nclaims = NULL, 
  premium = NULL 
)

Arguments
- df: data.frame with insurance portfolio
- x: column in df with risk factor
- severity: column in df with severity (default is NULL)
- exposure: column in df with exposure
- nclaims: column in df with number of claims
- premium: column in df with premium (default is NULL)

Value
An list of class univ_freq with components
- df: data frame with risk premium
- xvar: name of column in df with risk factor
- severity: name of column in df with severity
- nclaims: name of column in df with number of claims
- exposure: name of column in df with exposure
- premium: name of column in df with premium

Author(s)
Martin Haringa

Examples
univariate_risk_premium(MTPL2, x = area, severity = amount, exposure = exposure)
Index

*Topic datasets
  MTPL, 18
  MTPL2, 18

  autoplot.constructtariffclasses, 2
  autoplot.fitgam, 4
  autoplot.univ_all, 5
  autoplot.univ_avgpremium, 6
  autoplot.univ_avgsev, 7
  autoplot.univ_exposure, 8
  autoplot.univ_freq, 9
  autoplot.univ_lossratio, 10
  autoplot.univ_premium, 11

  biggest_reference, 12

  construct_tariff_classes, 13

  fisher, 15
  fit_gam, 16

  get_splits, 17

  MTPL, 18
  MTPL2, 18

  period_to_months, 19

  rating_factors, 20

  univariate_all, 21
  univariate_average_premium, 22
  univariate_average_severity, 23
  univariate_exposure, 24
  univariate_frequency, 25
  univariate_loss_ratio, 26
  univariate_risk_premium, 27