Package 'actuary'

May 7, 2026

Type Package

Title Actuarial Functions and Utilities

Version 0.1.2

Description Provides actuarial modeling tools for Monte Carlo loss simulations, loss reserving, and reinsurance layer loss calculations. It enables users to generate stochastic loss datasets with customisable frequency and severity distributions, fit development patterns to claim triangles, and calculate reinsurance losses for occurrence and aggregate layers with user-defined retentions, limits, and reinstatements. For development pattern selection, the package includes a machine learning approach that evaluates multiple reserving models using holdout validation to identify the best-fitting pattern based on predictive accuracy, this is based on the algorithm described in Richman, R and Balona, C (2020)<https://www.ssrn.com/abstract=3697256>.

License MIT + file LICENSE

BugReports https://github.com/andrewwilson201/actuary/issues

Depends R (>= 4.1.0)

Imports dplyr, dtplyr, forcats, furrr, future, gamlss.dist, ggplot2, ggtext, grDevices, MASS, pracma, progress, purrr, rlang, scales, stats, tidyr, yardstick

Suggests spelling

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

Author Andrew Wilson [aut, cre, cph]

Maintainer Andrew Wilson <andrewm.wilson@gmail.com>

Repository CRAN

Date/Publication 2026-03-04 10:30:07 UTC

Contents

`create_simulations` *create dataset of monte carlo simulations*

Description

create monte carlo simulations for a given frequency and severity distribution

Usage

```
create_simulations(
  num_sims,
  mean_freq,
  sev_dist,
  sev_param1 = 1,
  sev_param2 = 1,
  risk_shift = 0,
  var_freq = mean_freq,
  label = NULL,
  empirical = NULL
)
```

Arguments

`num_sims`	the number of simulated years required in the output
`mean_freq`	the mean number of claims in each simulated year. unless var_freq is provided assumes poisson
`sev_dist`	the severity distribution to use. one of lognorm2, gamma, pareto2, weibull or empirical
`sev_param1`	the first parameter for the severity distribution. not used if severity distribution is empirical. defaults to 1. for gamma and weibull this should be shape, for pareto2 it should be scale and for lognorm2 it should be meanlog.
`sev_param2`	the second parameter for the severity distribution. not used if severity distribution is empirical. defaults to 1. for gamma and weibull this should be scale, for pareto2 it should be shape and for lognorm2 it should be sdlog.

risk_shift	the amount to shift the simulated losses. defaults to zero
var_freq	the variance of the number of claims in each simulated year. if provided and different to mean freq will assume negative binomial. if provided and zero then the number of losses in each year will equal mean_freq
label	optional text variable to include as a column titled "label" in the output. see examples below.
empirical	if empirical distribution selected then this must be provided. dataset with two columns - probability and loss - representing the empirical distribution. probability must be cumulative probability and loss the associated loss amount.

Value

dataset of monte carlo simulations

Examples

```
# create 10k simulated years with poisson mean of 5 and
# lognormal parameters of 13 and 0.5
create_simulations(num_sims = 10e5,
                   mean_freq = 5,
                   sev_dist = "lognorm2",
                   sev_param1 = 13,
                   sev_param2 = 0.5)

# create 10k simulated years with sames as above except for
# negative binomial distribution with mean 5 and variance 10
create_simulations(num_sims = 10e5,
                   mean_freq = 5,
                   sev_dist = "lognorm2",
                   sev_param1 = 13,
                   sev_param2 = 0.5,
                   var_freq = 10)

# create 10k simulated years with two different distriubtions using pmap.
# use the label variable to differentiate them in the output
# note pmap doesn't work if an empirical severity distribution is being used
params <- dplyr::tibble(num_sims = 10e5,
                        mean_freq = c(5, 2),
                        sev_dist = c("lognorm2", "lognorm2"),
                        sev_param1 = c(13, 6),
                        sev_param2 = c(0.5, 0.7),
                        label = c("typeA", "typeB"))
purrr::pmap_dfr(params, create_simulations)

# create 10k simulated years with poisson frequency and empirical severity
# distribution where the loss amounts are distributed evenly between 5-10m.
# in practice you would provide a full CDF stored in a separate dataframe.
create_simulations(num_sims = 10e5,
                   mean_freq = 5,
                   sev_dist = "empirical",
```

```
                    empirical = dplyr::tibble(probability = c(0, 1),
                                              loss = c(5e6, 10e6)))
```

`fit_development_pattern` *fit chain ladder development pattern*

Description

requires tidy data. one row per cell in the unprojected triangle.
missing rows within the triangle will be filled in with the value in the previous period.
no rows for development periods beyond the last evaluation point.
data provided must be cumulative.
use the plot_development_pattern function to view fitted pattern.

Usage

```
fit_development_pattern(
  cohort_var,
  dev_var,
  weighting_var,
  data,
  dev_period_length = 1,
  dev_period_units = 12,
  exclude_last_diag = FALSE,
  num_periods = NULL,
  selected_curve = "weibull",
  smooth_from = NULL,
  future_dev_periods = 0,
  exclude_points = NULL,
  cohort_start = NULL,
  tail_factor = 1,
  exclude_high = FALSE,
  exclude_low = FALSE,
  bf_prior = NULL,
  cc_decay_factor = NULL,
  premium = NULL,
  calculate_mack_se = FALSE
)
```

Arguments

`cohort_var`	the variable in the input data representing the cohort

`dev_var` the variable in the input data representing the development period. needs to be consistent with dev_period_length and dev_period_units. e.g. if have half annual dev periods and data is in months (6, 12, 18 etc) then dev_period_length = 6 and dev_period_units = 1. if data in years (0.5, 1, 1.5 etc) then dev_period_length = 0.5 and dev_period_units = 12.

`weighting_var` the variable in the input data representing the amounts in the triangle e.g. claim counts or claim amounts etc

`data` the cumulative tidy dataset with the triangle data. one row per cohort year and development period. no rows for development periods beyond the last evaluation point.

`dev_period_length`
the difference between subsequent development periods in the units specified in dev_period_units

`dev_period_units`
the units in which dev_var is provided e.g. 1 for months, 12 for years etc.

`exclude_last_diag`
defaults to FALSE. set to TRUE to exclude last diagonal

`num_periods` the number of prior periods to include in the fit. defaults to the whole triangle but set to an integer to include the last n periods instead. if last diagonal is excluded it takes the n periods prior to these. if ratios are excluded due to any other parameter settings e.g. exclude_high then the number of periods is not adjusted to allow for this e.g. if 4 most recent rows are excluded via exclude_points and num_periods is set to 5 there will only be 1 row included in the ata factor calculation.

`selected_curve` which curve to use. weibull, exponential_decay or inverse_power. defaults to weibull. only used if smooth_from is provided.

`smooth_from` optional development period to smooth from. if selected a curve is fitted to the CL ata factors and ata factors are selected from this curve beyond the development period specified. the tail factor will be estimated from this fitted curve.

`future_dev_periods`
how many future development years to use when calculating the tail factor with the fitted curve. defaults to 5 years but should be set to a value such that the ata factors in the tail don't look to be truncated. only has an impact on the atu if smooth_from is provided. set to zero if you want to use a fitted curve but don't want it to estimate the tail.

`exclude_points` optional exclude individual ratios. provide as a list of indices into the ratio_table. column index should start after the cohort year column.

`cohort_start` all cohorts greater than or equal to this are included in the fit. defaults to the whole triangle if not specified.

`tail_factor` tail factor to apply. selected ata factor is set to this value at the final development period in the original data before future_dev_periods are added. hence be wary of using this in addition to using fitted curves to estimate the tail.

`exclude_high` set to 1 to exclude the highest ata ratio in each development period.

`exclude_low` set to 1 to exclude the lowest ata ratio in each development period.

bf_prior — optional bf prior to use when calculating ultimate. can't be supplied if cc_decay_factor is provided.

cc_decay_factor — optional cape cod decay factor to use when calculating ultimate. can't be supplied if bf_prior is provided.

premium — required if bf_prior or cc_decay_factor is provided. dataframe with with a column called premium and another column which must have the same name as cohort_var.

calculate_mack_se — if set to TRUE then return the Mack estimates of the standard error of the reserves in the ultimates dataframe

Value

a list with the fitted pattern, the original input data, a triangle of individual ratios, the r_squared for the curve fits, a monthly pattern, the development units and the projected ultimates. monthly pattern is fitted using a piecewise cubic Hermite interpolating polynomial and if using BF or CC is calculated using the BF / CC ultimates.

References

Mack, T (1993). Distribution-free calculation of the standard error of chain ladder reserve estimates. https://www.actuaries.org/LIBRARY/ASTIN/vol23no2/213.pdf

Examples

```
# fit chain ladder to example triangle data
fit_development_pattern(uw_year, dev_year, claim_number, triangle_data)

# exclude last diagonal, use a weibull fit to smooth from dev period 4
fit <- fit_development_pattern(uw_year,
                               dev_year,
                               claim_number,
                               triangle_data,
                               exclude_last_diag = TRUE,
                               smooth_from = 4)

# view the fitted pattern
plot_development_pattern(fit)

# view the table of individual ratios
fit$ratio_table

# ratios from dev 1 to dev 2 in uw year 1 looks high. this is row 1,
#   column 1 in the ratio_table
# note that column indexing starts from dev_1 onwards (not uw_year)
# refit excluding these points
fit <- fit_development_pattern(uw_year,
                               dev_year,
                               claim_number,
                               triangle_data,
```

```
                                exclude_points = list(c(1, 1)))
```

`fit_development_pattern_ml` *fit multiple development patterns and assess goodness of fit*

Description

1. creates a holdout sample as the last n diagonals in the triangle data provided where n is specified via the holdout_size parameter.
2. fits development patterns to the rest of the data, one for each row in the params_data dataframe.
3. the parameters for each of the development patterns are as specified in the params_data dataframe.
4. calculates the expected incremental claim amounts in the next diagonal from the holdout sample based on the fitted development pattern.
5. calculates the goodness of fit of the fitted development pattern by comparing the expected incremental amounts to the actual incremental amounts.
6. six goodness of fit metrics are calculated : AvE, negative AvE, CDR (see reference document or details section below for details on both), huber_loss, rmse, mae
7. add the first diagonal from the holdout sample to the training data and repeat steps 2 to 6.
8. repeat until all diagonals from the holdout sample are exhausted and then calculate the average goodness of the goodness of fit metrics across all iterations.
9. the total reserve implied by the pattern is calculated as the difference between the ultimate and the claim amount at the latest diagonal in the holdout data.

Usage

```
fit_development_pattern_ml(
  cohort_var,
  dev_var,
  weighting_var,
  data,
  dev_period_length,
  dev_period_units,
  params_data,
  holdout_size = 2,
  bf_priors = NULL,
  decay_factors = NULL,
  exposure_base = NULL,
  eval_metric = "ave_score",
  num_cores = 1
)
```

Arguments

cohort_var	the variable in the input data representing the cohort.
dev_var	the variable in the input data representing the development period. needs to be consistent with dev_period_length and dev_period_units. e.g. if have half annual dev periods and data is in months (6, 12, 18 etc) then dev_period_length = 6 and dev_period_units = 1. if data in years (0.5, 1, 1.5 etc) then dev_period_length = 0.5 and dev_period_units = 12.
weighting_var	the variable in the input data representing the amounts in the triangle e.g. claim counts or claim amounts etc.
data	the cumulative tidy dataset with the triangle data. one row per cohort year and development period. no rows for development periods beyond the last evaluation point.
dev_period_length	the difference between subsequent development periods in the units specified in dev_period_units.
dev_period_units	the units in which dev_var is provided e.g. 1 for months, 12 for years etc.
params_data	a dataframe with all combinations of parameters required for fitting the development patterns. must contain columns with headings : exclude_last_diag, smooth_from, exclude_high, exclude_low, selected_curve, num_periods. set smooth_from to 999 if you don't want to use curves at all. See fit_development_pattern documentation for more detail on the meaning of these parameters.
holdout_size	the number of diagonals to holdout when fitting the development patterns. a
bf_priors	if provided the algorithm will also assess the goodness of fit using the BF method to calculate reserves. can provide a single value or a vector of possible priors.
decay_factors	if provided the algorithm will additionally assess the goodness of fit using the generalised Cape Cod method to set the reserves. can provide a single value or a vector of possible priors.
exposure_base	must be provided if bf_priors or decay_factors is provided. a dataframe with a column called exposure (probably the premium) and another column which must have the same name as cohort_var.
eval_metric	which goodness of fit metric to order the results dataframe and to select the best parameters for the best_fit object.
num_cores	number of cores to use for parallel processing. defaults to 1.

Details

requires tidy data. one row per cell in the unprojected triangle.
missing rows within the triangle will be filled in with the value in the previous period.
no rows for development periods beyond the last evaluation point.
data provided must be cumulative.
AvE score is the weighted average of the AvE in each value of cohort_var where the weighting is the incremental value of weighting_var.
Negative AvE score is the negative unweighted average of the AvE. if using this as the eval_metric note that unlike for all other metrics, higher is better.

CDR score is the weighted average of the CDR in each value of cohort_var where the weighting is the incremental value of weighting_var.
CDR is the AvE plus the change in the IBNR estimate. equivalent to the change in projected ultimate claim value over the time period.

Value

1. a dataframe with the five goodness of fit metrics, the reserve and the parameter values used for the fit. ordered by descending value of eval_metric.
2. a fit_development_pattern object with the best fit according to the eval_metric.
3. a plot of reserve against eval_metric.
4. the exposure base if provided.
5. a tibble with the values of some of the inputs.

References

Richman, R and Balona, C (2020). The Actuary and IBNR Techniques: A Machine Learning Approach. https://ssrn.com/abstract=3697256

See Also

fit_development_pattern

Examples

```
# set up parameters file
cl_parameters <- tidyr::expand_grid(smooth_from = c(1, 2),
                                    exclude_last_diag = c(TRUE, FALSE),
                                    exclude_high = c(TRUE, FALSE),
                                    exclude_low = c(TRUE, FALSE),
                                    selected_curve = c("weibull",
                                                       "inverse_power",
                                                       "exponential_decay"),
                                    num_periods = c(1:5),
                                    future_dev_periods = c(0, 25))

# run function
ml_result <- fit_development_pattern_ml(uw_year,
                                        dev_year,
                                        claim_number,
                                        triangle_data, 1, 12,
                                        cl_parameters)

# view results
ml_result$results

# plot the best pattern according to the algorithm
plot_development_pattern(ml_result$best_fit)
```

```
# view the plot of reserve against the goodness of fit metric
ml_result$results_plot

# amend the plot so that selected curve is used for the colour of the points
ml_result$results_plot +
    ggplot2::geom_point(ggplot2::aes(colour = selected_curve))
```

`layer_loss` *calculate occurrence or aggregate layer losses*

Description

create a column with occurrence or aggregate losses into a layer with specified limit, retention and reinstatements

Usage

```
layer_loss(
  year,
  loss,
  retention,
  limit,
  reinstatements = 999,
  type = "occurrence"
)
```

Arguments

`year`	the variable in the input data that represents simulation year
`loss`	the variable in the input data that represents the loss amounts that will feed into the layer
`retention`	the retention the layer being priced
`limit`	the limit of the layer being priced
`reinstatements`	the number of reinstatements. defaults to 999
`type`	type of layer. occurence or aggregate. defaults to occurrence.

Value

losses into the layer

Examples

```
# calculate 10m x 10m layer with 1 reinstatement
losses |> dplyr::mutate(l1 = layer_loss(year, amount, 10e6, 10e6, 1))

# calculate 10m xs 10m agg layer with 0 reinstatements
losses |> dplyr::mutate(l1 = layer_loss(year, amount, 10e6, 10e6, 0, type = "aggregate"))
```

---

losses *example simulated loss data*

Description

monte carlo simulations with 20k simulated years. based on a poisson frequency distribution and lognormal severity distribution

Usage

```
losses
```

Format

A data frame with 49,936 rows and 3 columns:

year simulated year

amount ground-up simulated loss amount

territory_peril the territory peril in which the loss happened ...

Source

<data_raw/create example losses.R>

---

ml_pattern_range *get range of development patterns from fit_development_pattern_ml function*

Description

get range of development patterns from fit_development_pattern_ml function

Usage

```
ml_pattern_range(ml_results, num_fits = 25, metric = ave_score)
```

Arguments

ml_results	the list object returned by running the fit_development_pattern_ml
num_fits	the number of fits to include in the range. e.g. if set to 25 it will take the best 25 fitting patterns from the ml_results list object.
metric	which goodness of fit metric to use when selecting the best fits. ave_score, cdr_score, huber_loss, rmse, mae or neg_ave_score.

Value

a list with a plot showing the range of percentage developed at each development month and a tibble with the monthly pattern for each of the fits.

Examples

```
# set up parameters file
cl_parameters <- tidyr::expand_grid(smooth_from = c(1, 2),
                                    exclude_last_diag = c(TRUE, FALSE),
                                    exclude_high = c(TRUE, FALSE),
                                    exclude_low = c(TRUE, FALSE),
                                    selected_curve = c("weibull",
                                                       "inverse_power",
                                                       "exponential_decay"),
                                    num_periods = c(1:5),
                                    future_dev_periods = c(0, 25))

# run function
ml_result <- fit_development_pattern_ml(uw_year,
                                        dev_year,
                                        claim_number,
                                        triangle_data, 1, 12,
                                        cl_parameters)

# get range of top 25 best fitting patterns
ml_pattern_range(ml_result, 25)
```

---

plot_development_pattern

plot a develoment pattern fitted using the fit_development_pattern function

---

Description

provide the whole object returned by the fit_development_pattern not just the development_data item

Usage

```
plot_development_pattern(
  development_data,
  show_legend = FALSE,
  include_monthly_fit = FALSE,
  include_fitted_curve = FALSE
)
```

Arguments

development_data	the object returned by the fit_development_pattern function
show_legend	defaults to FALSE. show the legend for the colour variable in the plot
include_monthly_fit	include a separate graph showing the fitted monthly pattern against the chain ladder pattern. defaults to FALSE.
include_fitted_curve	defaults to FALSE. show the Weibull curve fit on the graph

Value

ggplot2 plot showing the fitted development pattern. if include_monthly_fit set to TRUE then returns a list containing two plots.

Examples

```
# fit chain ladder to example triangle data
fit <- fit_development_pattern(uw_year,
                               dev_year,
                               claim_number,
                               triangle_data)

# view the fitted pattern
plot_development_pattern(fit)

# view the fitted pattern and include the fitted monthly pattern in a
# separate graph
plot_development_pattern(fit, include_monthly_fit = TRUE)

# view the fitted pattern and include the Weibull curve fit
plot_development_pattern(fit, include_fitted_curve = TRUE)
```

`triangle_data` *example triangle data*

Description

tidy example triangle data. one row per development year and cohort. no development periods beyond the latest evaluation.

Usage

```
triangle_data
```

Format

A dataframe with 55 rows and 3 columns

uw_year underwriting year

dev_year development year

claim_number number of claims ...

Source

<data_raw/create_triangle_data.R>

Index