Package: lossdevtapp (via r-universe)

August 30, 2024

Title Actuarial Loss Development

Version 0.1.0

Description Actuarial Loss Development.

License MIT + file LICENSE

URL https://github.com/jimbrig/lossdevtapp,
https://jimbrig.github.io/lossdevtapp/

BugReports https://github.com/jimbrig/lossdevtapp/issues

Depends R (>= 4.1)

Imports checkmate, cli, config (>= 0.3.1), crayon, dplyr, DT, fs, glue, golem (>= 0.3.2), here, lubridate, magrittr, pkgload, purrr, randomNames, rlang, shiny (>= 1.7.1), shinycssloaders, shinydashboard, shinydashboardPlus, shinyjs, shinyWidgets, stats, tibble, tidyr, usethis, utils, waiter

Suggests actuar, devtools, fplot, kableExtra, knitr, rmarkdown, spelling, summarytools, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.2

Repository https://jimbrig.r-universe.dev

RemoteUrl https://github.com/jimbrig/lossdevtapp

RemoteRef HEAD

RemoteSha ed80a7895e5226d4cf9b129f419685c6136e6bd3

Contents

`aggregate_loss_data` *Aggregate Loss data*

Description

Aggregate Loss data

Usage

```
aggregate_loss_data(claim_dat, limit = NA)
```

Arguments

`claim_dat`	claims data
`limit`	optional limit

Value

df

app_header *App Header*

Description

Shiny App Dashboard’s Header Function

Functions that build the Shiny App’s UI:

- ‘app_ui‘: Main UI function

- ‘app_header‘: wrapper around [shinydashboard::dashboardHeader()]

- ‘app_sidebar‘: wrapper around [shinydashboard::dashboardSidebar()]

- ‘app_body‘: wrapper around [shinydashboard::dashboardBody()]

Shiny App’s User Interface Function.

Usage

```
app_header(title = "Loss Development", ...)

app_ui(request, ...)
```

Arguments

`title` App title to be placed in the header, above the sidebar.

`...` Arguments passed on to `shinydashboardPlus::dashboardHeader`

- `titleWidth` The width of the title area. This must either be a number which specifies the width in pixels, or a string that specifies the width in CSS units.
- `disable` If `TRUE`, don’t display the header bar.
- `.list` An optional list containing items to put in the header. Same as the `...` arguments, but in list format. This can be useful when working with programmatically generated items.
- `leftUi` Items that will appear on the left part of the navbar. Should be wrapped in a tagList.
- `controlbarIcon` Customize the trigger icon of the right sidebar.
- `fixed` Whether the navbar is fixed-top or not. FALSE by default.

`request` Internal parameter for ‘shiny‘.

Value

a [shinydashboard::dashboardHeader()]

The user interface definition, without modifications or side effects.

See Also

[shiny::shinyUI], [shinydashboard::dashboardPage()]

app_sys *app_sys*

Description

Access files for the shiny app from the package installation directory.

Usage

```
app_sys(...)
```

Arguments

`...`	character vectors, specifying subdirectory and file(s) within your package. The default, none, returns the root of the app.

Details

# Note If you manually change your package name in the ‘DESCRIPTION‘, don’t forget to change it here too, and in the config file (‘inst/config.yml‘).

For a safer name change mechanism, use the ‘golem::set_golem_name()‘ function.

create_triangle_bundle *Create Triangle Bundle*

Description

Create a "bundle" of triangle related items from an input loss dataset. The function returns a list of class "triangle_bundle" and an attribute describing which metric the bundle describes (i.e. paid, reported, counts, etc.).

The resulting list contains:

- aggregated data filtered for ages of maturity from the ‘age_increment‘ argument

- triangle data derived off the aggregated data

- the actual spread out triangle

- age_to_age data

- the age_to_age spread out triangle

- averages (currently only straight and weighted)

- initial selections for the LDF’s and derived CDF’s.

Usage

```
create_triangle_bundle(
  loss_data,
  age_increment = 12,
  origin_col = "accident_year",
  age_col = "devt",
  value_col = "paid"
)
```

Arguments

`loss_data`	initial aggregated loss data as a ‘data.frame‘
`age_increment`	increment in months between subsequent maturity periods
`origin_col`, `age_col`, `value_col`	column names as strings

Value

list of class "triangle_bundle" with an added attribute describing which metric the bundle describes (i.e. paid, reported, counts, etc.)

Examples

```
# create default paid triangle bundle
tri_paid_bundle <- create_triangle_bundle(loss_data_all)

# check out the structure
str(tri_paid_bundle)

# derive a similar bundle for reported dollars and counts
tri_rept_bundle <- create_triangle_bundle(loss_data_all, value_col = "reported")
tri_cnts_bundle <- create_triangle_bundle(loss_data_all, value_col = "n_claims")
```

`date_utils` *Date Utility Functions*

Description

Date utility helpers for deriving start/end dates.

Usage

```
end_of_month(date)

beg_of_month(date)

start_of_month(date)
```

Arguments

date Character string or Date representing the date to manipulate.

Value

Returns the Start or End Date as a Date.

Examples

```
# end_of_month ------------------------------------------------------------

# character input
end_of_month("2020-08-13")

# date input
end_of_month(as.Date("2020-08-13"))


# beg_of_month ------------------------------------------------------------

# character input
beg_of_month("2020-08-13")

# date input
beg_of_month(as.Date("2020-08-13"))


# start_of_month ----------------------------------------------------------

# character input
start_of_month("2020-08-13")

# date input
start_of_month(as.Date("2020-08-13"))
```

derive_triangles *Derive Triangles*

Description

Derive Triangles

Usage

```
derive_triangles(
  loss_dat,
  type = c("paid", "reported", "case", "n_claims"),
  limit = NULL
)
```

Arguments

`loss_dat`	loss data
`type`	paid, reported, case, or n_claims
`limit`	optinal limit

Value

list of triangle data

`dev_tri` *Development Triangle Class*

Description

Development Triangle Class

Usage

```
dev_tri(origin, age, value, value_label = NULL, latest_eval_date = NULL)
```

Arguments

`origin`, `age`, `value`
columns necessary to generate a ‘dev_tri‘

`value_label` optional label for the values (i.e. paid, incurred)

`latest_eval_date`
optional - specify latest val date

Examples

```
library(lossdevtapp)

my_triangle <- dev_tri(
  origin = loss_data$accident_year,
  age = loss_data$devt,
  value = loss_data$payment,
  value_label = "paid",
  latest_eval_date = max(loss_data$eval_date)
)

class(my_triangle)
str(my_triangle)
print(my_triangle)
```

doc_data *Document Datasets*

Description

Creates skeleton to document datasets via ‘roxygen2‘.

Usage

```
doc_data(
  obj,
  title = deparse(substitute(obj)),
  description = "DATASET_DESCRIPTION",
  write_to_file = TRUE,
  ...
)
```

Arguments

`obj`	object to document
`title`	Title
`description`	Description
`write_to_file`	Logical
`...`	N/A

Value

silently returns the doc_string

Examples

```
data("loss_data")

string <- doc_data(losses, "Loss Data", "Claims Data", FALSE)

cat(string)
```

`feedback` *Feedback Message Utility Functions*

Description

A set of helper functions for providing verbose feedback to the developer using this packages functions.

Usage

```
msg_field(x)

msg_value(x)

msg_done(x)

msg_bullet(x, bullet = cli::symbol$bullet)

msg_err(x)

msg_path(x)

msg_info(x)

msg_code(x)

msg_feedback(x)
```

Arguments

`x`	The string passed to various ‘msg_‘ functions.
`bullet`	What to use for the message’s ‘bullet‘. Defaults to ‘cli::symbol$bullet‘

See Also

- [usethis::ui-questions()] - [cli::list_symbols()]

Other Feedback Utilities: `indent()`, `inform()`

indent *Indent*

Description

Indentation around various ‘msg_‘ feedback functions.

Usage

```
indent(x, first = "  ", indent = first)
```

Arguments

x The string passed to various ‘msg_‘ functions.

first what to indent with - defaults to ‘" "‘.

indent indentation of next line - defaults to ‘first‘

Value

string

See Also

Other Feedback Utilities: feedback, inform()

inform *Inform*

Description

A wrapper around [rlang::inform()] for providing feedback to developers using this packages functions.

Usage

```
inform(...)
```

Arguments

`...` Arguments passed on to `rlang::inform`

`message` The message to display, formatted as a **bulleted list**. The first element is displayed as an *alert* bullet prefixed with `!` by default. Elements named `"*"`, `"i"`, `"v"`, `"x"`, and `"!"` are formatted as regular, info, success, failure, and error bullets respectively. SeFormatting messages with cli)` for more about bulleted messaging.

If a message is not supplied, it is expected that the message is generated **lazily** through `cnd_header()` and `cnd_body()` methods. In that case, `class` must be supplied. Only `inform()` allows empty messages as it is occasionally useful to build user output incrementally.

If a function, it is stored in the `header` field of the error condition. This acts as a `cnd_header()` method that is invoked lazily when the error message is displayed.

`class` Subclass of the condition.

`body,footer` Additional bullets.

`use_cli_format` Whether to format `message` lazily using cli if available. This results in prettier and more accurate formatting of messages. See `local_use_cli()` to set this condition field by default in your package namespace.

If set to `TRUE`, `message` should be a character vector of individual and unformatted lines. Any newline character `"\\n"` already present in `message` is reformatted by cli's paragraph formatter. See Formatting messages with cli.

`.file` A connection or a string specifying where to print the message. The default depends on the context, see the `stdout` vs `stderr` section.

`.subclass` **[Deprecated]** This argument was renamed to `class` in rlang 0.4.2 for consistency with our conventions for class constructors documented in `https://adv-r.hadley.nz/s3.html#s3-subclassing`.

`.frequency` How frequently should the warning or message be displayed? By default (`"always"`) it is displayed at each time. If `"regularly"`, it is displayed once every 8 hours. If `"once"`, it is displayed once per session.

`.frequency_id` A unique identifier for the warning or message. This is used when `.frequency` is supplied to recognise recurring conditions. This argument must be supplied if `.frequency` is not set to `"always"`.

Value

feedback in console

See Also

[rlang::inform()]

Other Feedback Utilities: `feedback`, `indent()`

loss_data *loss_data*

Description

Actuarial claims loss data.

Usage

```
loss_data
```

Format

A data frame with 70331 rows and 22 variables:

`eval_date` double. DESCRIPTION.

`claim_num` integer. DESCRIPTION.

`claim_id` character. DESCRIPTION.

`accident_date` double. DESCRIPTION.

`state` character. DESCRIPTION.

`claimant` character. DESCRIPTION.

`report_date` double. DESCRIPTION.

`status` character. DESCRIPTION.

`payment` double. DESCRIPTION.

`case` double. DESCRIPTION.

`transaction_date` double. DESCRIPTION.

`trans_num` integer. DESCRIPTION.

`paid` double. DESCRIPTION.

`reported` double. DESCRIPTION.

`accident_year` double. DESCRIPTION.

`report_year` double. DESCRIPTION.

`eval_year` double. DESCRIPTION.

`ay_start` double. DESCRIPTION.

`ay_end` double. DESCRIPTION.

`ay_avg` double. DESCRIPTION.

`devt_in_days` double. DESCRIPTION.

`devt` double. DESCRIPTION.

`loss_data_all` *loss_data_all*

Description

Actuarial claims loss data.

Usage

```
loss_data_all
```

Format

A data frame with 888864 rows and 22 variables:

`eval_date` double. DESCRIPTION.

`claim_num` integer. DESCRIPTION.

`claim_id` character. DESCRIPTION.

`accident_date` double. DESCRIPTION.

`state` character. DESCRIPTION.

`claimant` character. DESCRIPTION.

`report_date` double. DESCRIPTION.

`status` character. DESCRIPTION.

`payment` double. DESCRIPTION.

`case` double. DESCRIPTION.

`transaction_date` double. DESCRIPTION.

`trans_num` integer. DESCRIPTION.

`paid` double. DESCRIPTION.

`reported` double. DESCRIPTION.

`accident_year` double. DESCRIPTION.

`report_year` double. DESCRIPTION.

`eval_year` double. DESCRIPTION.

`ay_start` double. DESCRIPTION.

`ay_end` double. DESCRIPTION.

`ay_avg` double. DESCRIPTION.

`devt_in_days` double. DESCRIPTION.

`devt` double. DESCRIPTION.

run_app *Run the Shiny Application*

Description

Run the Shiny Application

Usage

```
run_app(
  onStart = NULL,
  options = list(),
  enableBookmarking = NULL,
  uiPattern = "/",
  ...
)
```

Arguments

`onStart` A function that will be called before the app is actually run. This is only needed for `shinyAppObj`, since in the `shinyAppDir` case, a `global.R` file can be used for this purpose.

`options` Named options that should be passed to the `runApp` call (these can be any of the following: "port", "launch.browser", "host", "quiet", "display.mode" and "test.mode"). You can also specify `width` and `height` parameters which provide a hint to the embedding environment about the ideal height/width for the app.

`enableBookmarking` Can be one of `"url"`, `"server"`, or `"disable"`. The default value, `NULL`, will respect the setting from any previous calls to `enableBookmarking()`. See `enableBookmarking()` for more information on bookmarking your app.

`uiPattern` A regular expression that will be applied to each `GET` request to determine whether the `ui` should be used to handle the request. Note that the entire request path must match the regular expression in order for the match to be considered successful.

`...` arguments to pass to golem_opts. See ‘?golem::get_golem_options‘ for more details.

simulate_claims *simulate_claims*

Description

A function to simulate *transactional* actuarial claims/loss data for Property Casualty Insurance.

Usage

```
simulate_claims(
  n_claims = 1000,
  start_date = "2015-01-01",
  end_date = Sys.Date(),
  seed = 12345,
  loss_distribution = "lnorm",
  params = list(mean_log = 7.5, sd_log = 1.5),
  status_prob_open = 0.96,
  cache = FALSE,
  ...
)
```

Arguments

`n_claims` Numeric - Number of claims to be simulated.

`start_date, end_date`
Character/Date - Start and End dates for simulation to create claims within (experience_period).

`seed` Numeric - the seed is used to isolate randomness during statistical simulations.

`loss_distribution`
Character - must be one of the distributions mentioned in the details below. Defaults to lognormal.

`params` Parameters associated with the specified ‘loss_distribution‘ in a list (i.e. ‘list(mean_log = 7.5, sd_log = 1.5)‘ for lognormal distribution).

`status_prob_open`
Numeric - must be within ‘0 < x < 1‘ and represents probability a claim is open when running binomial simulations for claims’ status.

`cache` Boolean/Logical - enable caching?

`...` If needed

Details

Severity/Loss Distributions: - Normal: ‘norm‘ - Parameters are ‘mean‘ and ‘sd‘. - Lognormal: ‘lnorm‘ - Parameters are ‘meanlog‘ and ‘sdlog‘. - Gamma: ‘gamma‘ - Shape, Rate, Scale - LogGamma: ‘lgamma‘ - Shapelog, Ratelog - Pareto: ‘pareto‘ - Shape and Scale - Weibull: ‘weibull‘ - Shape and Scale - Generalized Beta: ‘genbeta‘ - Shape1, Shape2, Shape3, Rate, Scale

Value

The return value, if any, from executing the function.

triangles_module *Triangles Module*

Description

Shiny module containing ‘mod_triangles_ui‘ and ‘mod_triangles_server‘, respectively.

This module renders a user interface for displaying and analyzing actuarial loss data in the form of loss development triangles.

A shiny Module.

Usage

```
mod_triangles_ui(id, loss_data = loss_data_all)

mod_triangles_server(id, loss_data, selected_eval)
```

Arguments

id	ID associated with UI counterpart
loss_data	loss data
selected_eval	selected evaluation date

Index