Package 'actxps'

September 24, 2025

Title Create Actuarial Experience Studies: Prepare Data, Summarize Results, and Create Reports

Version 1.6.1

Maintainer Matt Heaphy <mattrmattrs@gmail.com>

Description Experience studies are used by actuaries to explore historical experience across blocks of business and to inform assumption setting activities. This package provides functions for preparing data, creating studies, visualizing results, and beginning assumption development. Experience study methods, including exposure calculations, are described in: Atkinson & McGarry (2016) ``Experience Study Calculations'' <https://www.soa.org/49378a/globalassets/assets/files/research/experience-study-calculations.pdf>. The limited fluctuation credibility method used by the 'exp_stats()' function is described in: Herzog (1999, ISBN:1-56698-374-6) ``Introduction to Credibility Theory''.

License MIT + file LICENSE

URL https://github.com/mattheaphy/actxps/, https://mattheaphy.github.io/actxps/

BugReports https://github.com/mattheaphy/actxps/issues

Encoding UTF-8

RoxygenNote 7.3.2

Suggests knitr, RColorBrewer, rmarkdown, testthat (>= 3.0.0), shiny (>= 1.7.5), bslib (>= 0.9.0), thematic

Config/testthat/edition 3

Depends R (>= 4.1)

Imports dplyr (>= 1.1.1), ggplot2, tibble, rlang, glue, purrr, scales, gt (>= 0.9.0), paletteer, recipes, generics, readr, tidyr, vctrs, clock, cli

LazyData true

VignetteBuilder knitr

NeedsCompilation no

Author Matt Heaphy [aut, cre]

Repository CRAN

Date/Publication 2025-09-24 13:40:02 UTC

Contents

add_predictions . . . 2
add_transactions . . . 3
agg_sim_dat . . . 4
as_exp_df . . . 5
as_trx_df . . . 8
autoplot_exp . . . 11
autotable . . . 13
expose . . . 15
expose_split . . . 18
exp_shiny . . . 19
exp_stats . . . 22
is_exposed_df . . . 25
plot_special . . . 27
plot_special_trx . . . 28
pol_yr . . . 29
qx_iamb . . . 30
sim_data . . . 31
step_expose . . . 32
summary.exposed_df . . . 33
toy_census . . . 34
trx_stats . . . 35

Index **39**

`add_predictions` *Add predictions to a data frame*

Description

Attach predicted values from a model to a data frame with exposure-level records.

Usage

```
add_predictions(.data, model, ..., col_expected = NULL)
```

Arguments

.data	A data frame, preferably with the class exposed_df
model	A model object that has an S3 method for predict()
...	Additional arguments passed to predict()
col_expected	NULL or a character vector containing column names for each value returned by predict()

Details

This function attaches predictions from a model to a data frame that preferably has the class exposed_df. The model argument must be a model object that has an S3 method for the predict() function. This method must have new data for predictions as the second argument.

The col_expected argument is optional.

- If NULL, names from the result of predict() will be used. If there are no names, a default name of "expected" is assumed. In the event that predict() returns multiple values, the default name will be suffixed by "_x", where x = 1 to the number of values returned.
- If a value is passed, it must be a character vector of same length as the result of predict()

Value

A data frame or exposed_df object with one of more new columns containing predictions.

Examples

```
expo <- expose_py(census_dat, "2019-12-31") |>
  mutate(surrender = status == "Surrender")
mod <- glm(surrender ~ inc_guar + pol_yr, expo, family = 'binomial')
add_predictions(expo, mod, type = 'response')
```

---

add_transactions *Add transactions to an experience study*

Description

Attach summarized transactions to a data frame with exposure-level records.

Usage

```
add_transactions(
  .data,
  trx_data,
  col_pol_num = "pol_num",
  col_trx_date = "trx_date",
  col_trx_type = "trx_type",
  col_trx_amt = "trx_amt"
)
```

Arguments

.data	A data frame with exposure-level records with the class exposed_df. Use as_exposed_df() to convert a data frame to an exposed_df object if necessary.
trx_data	A data frame containing transactions details. This data frame must have columns for policy numbers, transaction dates, transaction types, and transaction amounts.
col_pol_num	Name of the column in trx_data containing the policy number
col_trx_date	Name of the column in trx_data containing the transaction date
col_trx_type	Name of the column in trx_data containing the transaction type
col_trx_amt	Name of the column in trx_data containing the transaction amount

Details

This function attaches transactions to an exposed_df object. Transactions are grouped and summarized such that the number of rows in the exposed_df object does not change. Two columns are added to the output for each transaction type. These columns have names of the pattern trx_n_{*} (transaction counts) and trx_amt_{*} (transaction_amounts).

Transactions are associated with the exposed_df object by matching transactions dates with exposure dates ranges found in exposed_df.

All columns containing dates must be in YYYY-MM-DD format.

Value

An exposed_df object with two new columns containing transaction counts and amounts for each transaction type found in trx_data. The exposed_df's trx_types attributes will be updated to include the new transaction types found in trx_data.

See Also

expose(), as_exposed_df()

Examples

```
expo <- expose_py(census_dat, "2019-12-31", target_status = "Surrender")
add_transactions(expo, withdrawals)
```

---

agg_sim_dat *Aggregate simulated annuity data*

Description

A pre-aggregated version of surrender and withdrawal experience from the simulated data sets census_dat, withdrawals, and account_vals. This data is theoretical only and does not represent the experience on any specific product.

Usage

```
agg_sim_dat
```

Format

A data frame containing summarized experience study results grouped by policy year, income guarantee presence, tax-qualified status, and product.

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 180 rows and 16 columns.

Details

pol_yr Policy year

inc_guar Indicates whether the policy was issued with an income guarantee

qual Indicates whether the policy was purchased with tax-qualified funds

product Product: a, b, or c

exposure_n Sum of policy year exposures by count

claims_n Sum of claim counts

av Sum of account value

exposure_amt Sum of policy year exposures weighted by account value

claims_amt Sum of claims weighted by account value

av_sq Sum of squared account values

n Number of exposure records

wd Sum of partial withdrawal transactions

wd_n Count of partial withdrawal transactions

wd_flag Count of exposure records with partial withdrawal transactions

wd_sq Sum of squared partial withdrawal transactions

av_w_wd Sum of account value for exposure records with partial withdrawal transactions

See Also

census_dat

---

as_exp_df *Termination summary helper functions*

Description

Convert aggregate termination experience studies to the `exp_df` class.

Usage

```
as_exp_df(
  x,
  expected = NULL,
  wt = NULL,
  col_claims,
  col_exposure,
  col_n_claims,
  col_weight_sq,
  col_weight_n,
  target_status = NULL,
  start_date = as.Date("1900-01-01"),
  end_date = NULL,
  credibility = FALSE,
  conf_level = 0.95,
  cred_r = 0.05,
  conf_int = FALSE
)

is_exp_df(x)
```

Arguments

x	An object. For as_exp_df(), x must be a data frame.
expected	A character vector containing column names in x with expected values
wt	Optional. Length 1 character vector. Name of the column in x containing weights to use in the calculation of claims, exposures, partial credibility, and confidence intervals.
col_claims	Optional. Name of the column in x containing claims. The assumed default is "claims".
col_exposure	Optional. Name of the column in x containing exposures. The assumed default is "exposure".
col_n_claims	Optional and only used used when wt is passed. Name of the column in x containing the number of claims.
col_weight_sq	Optional and only used used when wt is passed. Name of the column in x containing the sum of squared weights.
col_weight_n	Optional and only used used when wt is passed. Name of the column in x containing exposure record counts.
target_status	Character vector of target status values. Default value = NULL.
start_date	Experience study start date. Default value = 1900-01-01.
end_date	Experience study end date
credibility	If TRUE, future calls to summary() will include partial credibility weights and credibility-weighted termination rates.
conf_level	Confidence level used for the Limited Fluctuation credibility method and confidence intervals

cred_r	Error tolerance under the Limited Fluctuation credibility method
conf_int	If TRUE, future calls to summary() will include confidence intervals around the observed termination rates and any actual-to-expected ratios.

Details

is_exp_df() will return TRUE if x is an exp_df object.

as_exp_df() will coerce a data frame to an exp_df object if that data frame has columns for exposures and claims.

as_exp_df() is most useful for working with aggregate summaries of experience that were not created by actxps where individual policy information is not available. After converting the data to the exp_df class, summary() can be used to summarize data by any grouping variables, and autoplot() and autotable() are available for reporting.

If nothing is passed to wt, the data frame x must include columns containing:

- Exposures (exposure)
- Claim counts (claims)

If wt is passed, the data must include columns containing:

- Weighted exposures (exposure)
- Weighted claims (claims)
- Claim counts (n_claims)
- The raw sum of weights **NOT** multiplied by exposures
- Exposure record counts (.weight_n)
- The raw sum of squared weights (.weight_sq)

The names in parentheses above are expected column names. If the data frame passed to as_exp_df() uses different column names, these can be specified using the col_* arguments.

When a column name is passed to wt, the columns .weight, .weight_n, and .weight_sq are used to calculate credibility and confidence intervals. If credibility and confidence intervals aren't required, then it is not necessary to pass anything to wt. The results of as_exp_df() and any downstream summaries will still be weighted as long as the exposures and claims are pre-weighted.

target_status, start_date, and end_date are optional arguments that are only used for printing the resulting exp_df object.

Value

For is_exp_df(), a length-1 logical vector. For as_exp_df(), an exp_df object.

See Also

exp_stats() for information on how exp_df objects are typically created from individual exposure records.

Examples

```
# convert pre-aggregated experience into an exp_df object
dat <- as_exp_df(agg_sim_dat, col_exposure = "exposure_n",
                 col_claims = "claims_n",
                 target_status = "Surrender",
                 start_date = 2005, end_date = 2019,
                 conf_int = TRUE)
dat
is_exp_df(dat)

# summary by policy year
summary(dat, pol_yr)

# repeat the prior exercise on a weighted basis
dat_wt <- as_exp_df(agg_sim_dat, wt = "av",
                    col_exposure = "exposure_amt",
                    col_claims = "claims_amt",
                    col_n_claims = "claims_n",
                    col_weight_sq = "av_sq",
                    col_weight_n = "n",
                    target_status = "Surrender",
                    start_date = 2005, end_date = 2019,
                    conf_int = TRUE)
dat_wt

# summary by policy year
summary(dat_wt, pol_yr)
```

---

as_trx_df *Transaction summary helper functions*

Description

Convert aggregate transaction experience studies to the `trx_df` class.

Usage

```
as_trx_df(
  x,
  col_trx_amt = "trx_amt",
  col_trx_n = "trx_n",
  col_trx_flag = "trx_flag",
  col_exposure = "exposure",
  col_percent_of = NULL,
  col_percent_of_w_trx = NULL,
  col_trx_amt_sq = "trx_amt_sq",
  start_date = as.Date("1900-01-01"),
```

```
  end_date = NULL,
  conf_int = FALSE,
  conf_level = 0.95
)

is_trx_df(x)
```

Arguments

x	An object. For as_trx_df(), x must be a data frame.
col_trx_amt	Optional. Name of the column in x containing transaction amounts.
col_trx_n	Optional. Name of the column in x containing transaction counts.
col_trx_flag	Optional. Name of the column in x containing the number of exposure records with transactions.
col_exposure	Optional. Name of the column in x containing exposures.
col_percent_of	Optional. Name of the column in x containing a numeric variable to use in "percent of" calculations.
col_percent_of_w_trx	Optional. Name of the column in x containing a numeric variable to use in "percent of" calculations with transactions.
col_trx_amt_sq	Optional and only required when col_percent_of is passed and conf_int is TRUE. Name of the column in x containing squared transaction amounts.
start_date	Experience study start date. Default value = 1900-01-01.
end_date	Experience study end date
conf_int	If TRUE, future calls to summary() will include confidence intervals around the observed utilization rates and any percent_of output columns.
conf_level	Confidence level for confidence intervals

Details

is_trx_df() will return TRUE if x is a trx_df object.

as_trx_df() will coerce a data frame to a trx_df object if that data frame has the required columns for transaction studies listed below.

as_trx_df() is most useful for working with aggregate summaries of experience that were not created by actxps where individual policy information is not available. After converting the data to the trx_df class, summary() can be used to summarize data by any grouping variables, and autoplot() and autotable() are available for reporting.

At a minimum, the following columns are required:

- Transaction amounts (trx_amt)
- Transaction counts (trx_n)
- The number of exposure records with transactions (trx_flag). This number is not necessarily equal to transaction counts. If multiple transactions are allowed per exposure period, trx_flag will be less than trx_n.

- Exposures (`exposure`)

If transaction amounts should be expressed as a percentage of another variable (i.e. to calculate utilization rates or actual-to-expected ratios), additional columns are required:

- A denominator "percent of" column. For example, the sum of account values.
- A denominator "percent of" column for exposure records with transactions. For example, the sum of account values across all records with non-zero transaction amounts.

If confidence intervals are desired and "percent of" columns are passed, an additional column for the sum of squared transaction amounts (`trx_amt_sq`) is also required.

The names in parentheses above are expected column names. If the data frame passed to `as_trx_df()` uses different column names, these can be specified using the `col_*` arguments.

`start_date`, and `end_date` are optional arguments that are only used for printing the resulting `trx_df` object.

Unlike `trx_stats()`, `as_trx_df()` only permits a single transaction type and a single `percent_of` column.

Value

For `is_trx_df()`, a length-1 logical vector. For `as_trx_df()`, a `trx_df` object.

See Also

`trx_stats()` for information on how `trx_df` objects are typically created from individual exposure records.

Examples

```
# convert pre-aggregated experience into a trx_df object
dat <- as_trx_df(agg_sim_dat,
                 col_exposure = "n",
                 col_trx_amt = "wd",
                 col_trx_n = "wd_n",
                 col_trx_flag = "wd_flag",
                 col_percent_of = "av",
                 col_percent_of_w_trx = "av_w_wd",
                 col_trx_amt_sq = "wd_sq",
                 start_date = 2005, end_date = 2019,
                 conf_int = TRUE)
dat
is_trx_df(dat)

# summary by policy year
summary(dat, pol_yr)
```

`autoplot_exp` *Plot experience study results*

Description

Plot experience study results

Usage

```
## S3 method for class 'exp_df'
autoplot(
  object,
  ...,
  x = NULL,
  y = NULL,
  color = NULL,
  mapping,
  second_axis = FALSE,
  second_y = NULL,
  scales = "fixed",
  geoms = c("lines", "bars", "points"),
  y_labels = scales::label_percent(accuracy = 0.1),
  second_y_labels = scales::label_comma(accuracy = 1),
  y_log10 = FALSE,
  conf_int_bars = FALSE
)

## S3 method for class 'trx_df'
autoplot(
  object,
  ...,
  x = NULL,
  y = NULL,
  color = NULL,
  mapping,
  second_axis = FALSE,
  second_y = NULL,
  scales = "fixed",
  geoms = c("lines", "bars", "points"),
  y_labels = scales::label_percent(accuracy = 0.1),
  second_y_labels = scales::label_comma(accuracy = 1),
  y_log10 = FALSE,
  conf_int_bars = FALSE
)
```

Arguments

object	An object of class exp_df created by the function exp_stats() or an object of class trx_df created by the function trx_stats().
...	Faceting variables passed to ggplot2::facet_wrap().
x	An unquoted column name in object or expression to use as the x variable.
y	An unquoted column name in object or expression to use as the y variable. If unspecified, y will default to the observed termination rate (q_obs) for exp_df objects and the observed utilization rate (trx_util) for trx_df objects.
color	An unquoted column name in object or expression to use as the color and fill variables.
mapping	Aesthetic mapping passed to ggplot2::ggplot(). NOTE: If mapping is supplied, the x, y, and color arguments will be ignored.
second_axis	Logical. If TRUE, the variable specified by second_y (default = exposure) is plotted on a second y-axis using an area geometry.
second_y	An unquoted column name in object to use as the y variable on the second y-axis. If unspecified, this will default to exposure.
scales	The scales argument passed to ggplot2::facet_wrap().
geoms	Type of geometry. If "lines" is passed, the plot will display lines and points. If "bars", the plot will display bars. If "points", the plot will display points only.
y_labels	Label function passed to ggplot2::scale_y_continuous().
second_y_labels	Same as y_labels, but for the second y-axis.
y_log10	If TRUE, the y-axes are plotted on a log-10 scale.
conf_int_bars	If TRUE, confidence interval error bars are included in the plot. For exp_df objects, this option is available for termination rates and actual-to-expected ratios. For trx_df objects, this option is available for utilization rates and any pct_of columns.

Details

If no aesthetic map is supplied, the plot will use the first grouping variable in object on the x axis and q_obs on the y axis. In addition, the second grouping variable in object will be used for color and fill.

If no faceting variables are supplied, the plot will use grouping variables 3 and up as facets. These variables are passed into ggplot2::facet_wrap(). Specific to trx_df objects, transaction type (trx_type) will also be added as a faceting variable.

Value

a ggplot object

See Also

plot_termination_rates(), plot_actual_to_expected()

Examples

```
study_py <- expose_py(census_dat, "2019-12-31", target_status = "Surrender")

study_py <- study_py |>
  add_transactions(withdrawals)

exp_res <- study_py |> group_by(pol_yr) |> exp_stats()
autoplot(exp_res)

trx_res <- study_py |> group_by(pol_yr) |> trx_stats()
autoplot(trx_res)
```

`autotable` *Tabular experience study summary*

Description

`autotable()` is a generic function used to create a table from an object of a particular class. Tables are constructed using the `gt` package.

`autotable.exp_df()` is used to convert experience study results to a presentation-friendly format.

`autotable.trx_df()` is used to convert transaction study results to a presentation-friendly format.

Usage

```
autotable(object, ...)

## S3 method for class 'exp_df'
autotable(
  object,
  fontsize = 100,
  decimals = 1,
  colorful = TRUE,
  color_q_obs = "RColorBrewer::GnBu",
  color_ae_ = "RColorBrewer::RdBu",
  rename_cols = rlang::list2(...),
  show_conf_int = FALSE,
  show_cred_adj = FALSE,
  decimals_amt = 0,
  suffix_amt = FALSE,
  show_total = FALSE,
  ...
)

## S3 method for class 'trx_df'
autotable(
  object,
```

```
  fontsize = 100,
  decimals = 1,
  colorful = TRUE,
  color_util = "RColorBrewer::GnBu",
  color_pct_of = "RColorBrewer::RdBu",
  rename_cols = rlang::list2(...),
  show_conf_int = FALSE,
  decimals_amt = 0,
  suffix_amt = FALSE,
  show_total = FALSE,
  ...
)
```

Arguments

object	An object of class exp_df usually created by the function exp_stats() or an object of class trx_df created by the trx_stats() function.
...	Additional arguments passed to gt::gt().
fontsize	Font size percentage multiplier.
decimals	Number of decimals to display for percentages
colorful	If TRUE, color will be added to the the observed termination rate and actual-to-expected columns for termination studies, and the utilization rate and "percentage of" columns for transaction studies.
color_q_obs	Color palette used for the observed termination rate.
color_ae_	Color palette used for actual-to-expected rates.
rename_cols	An optional list consisting of key-value pairs. This can be used to relabel columns on the output table. This parameter is most useful for renaming grouping variables that will appear under their original variable names if left unchanged. See gt::cols_label() for more information.
show_conf_int	If TRUE confidence intervals will be displayed assuming they are available on object.
show_cred_adj	If TRUE credibility-weighted termination rates will be displayed assuming they are available on object.
decimals_amt	Number of decimals to display for amount columns (number of claims, claim amounts, exposures, transaction counts, total transactions, and average transactions)
suffix_amt	This argument has the same meaning as the suffixing argument in gt::fmt_number() for amount columns. If FALSE (the default), no scaling or suffixing are applied to amount columns. If TRUE, all amount columns are automatically scaled and suffixed by "K" (thousands), "M" (millions), "B" (billions), or "T" (trillions). See gt::fmt_number() for more information.
show_total	If TRUE the table will include grand total row(s).
color_util	Color palette used for utilization rates.
color_pct_of	Color palette used for "percentage of" columns.

Details

The color_q_obs, color_ae_, color_util, and color_pct_of arguments must be strings referencing a discrete color palette available in the paletteer package. Palettes must be in the form "package::palette". For a full list of available palettes, see paletteer::palettes_d_names.

Value

a gt object

Examples

```
if (interactive()) {
  study_py <- expose_py(census_dat, "2019-12-31", target_status = "Surrender")
  expected_table <- c(seq(0.005, 0.03, length.out = 10), 0.2, 0.15, rep(0.05, 3))

  study_py <- study_py |>
    mutate(expected_1 = expected_table[pol_yr],
           expected_2 = ifelse(inc_guar, 0.015, 0.03)) |>
    add_transactions(withdrawals) |>
    left_join(account_vals, by = c("pol_num", "pol_date_yr"))

  exp_res <- study_py |> group_by(pol_yr) |>
    exp_stats(expected = c("expected_1", "expected_2"), credibility = TRUE,
              conf_int = TRUE)
  autotable(exp_res)

  trx_res <- study_py |> group_by(pol_yr) |>
    trx_stats(percent_of = "av_anniv", conf_int = TRUE)
  autotable(trx_res)
}
```

---

expose *Create exposure records from census records*

Description

Convert a data frame of census-level records to exposure-level records.

Usage

```
expose(
  .data,
  end_date,
  start_date = as.Date("1900-01-01"),
  target_status = NULL,
  cal_expo = FALSE,
  expo_length = c("year", "quarter", "month", "week"),
```

```
  col_pol_num = "pol_num",
  col_status = "status",
  col_issue_date = "issue_date",
  col_term_date = "term_date",
  default_status
)

expose_py(...)

expose_pq(...)

expose_pm(...)

expose_pw(...)

expose_cy(...)

expose_cq(...)

expose_cm(...)

expose_cw(...)
```

Arguments

`.data`	A data frame with census-level records
`end_date`	Experience study end date
`start_date`	Experience study start date. Default value = 1900-01-01.
`target_status`	Character vector of target status values. Default value = `NULL`.
`cal_expo`	Set to TRUE for calendar year exposures. Otherwise policy year exposures are assumed.
`expo_length`	Exposure period length
`col_pol_num`	Name of the column in `.data` containing the policy number
`col_status`	Name of the column in `.data` containing the policy status
`col_issue_date`	Name of the column in `.data` containing the issue date
`col_term_date`	Name of the column in `.data` containing the termination date
`default_status`	Optional scalar character representing the default active status code. If not provided, the most common status is assumed.
`...`	Arguments passed to `expose()`

Details

Census-level data refers to a data set wherein there is one row per unique policy. Exposure-level data expands census-level data such that there is one record per policy per observation period. Observation periods could be any meaningful period of time such as a policy year, policy month, calendar year, calendar quarter, calendar month, etc.

`target_status` is used in the calculation of exposures. The annual exposure method is applied, which allocates a full period of exposure for any statuses in `target_status`. For all other statuses, new entrants and exits are partially exposed based on the time elapsed in the observation period. This method is consistent with the Balducci Hypothesis, which assumes that the probability of termination is proportionate to the time elapsed in the observation period. If the annual exposure method isn't desired, `target_status` can be ignored. In this case, partial exposures are always applied regardless of status.

`default_status` is used to indicate the default active status that should be used when exposure records are created.

Value

A tibble with class `exposed_df`, `tbl_df`, `tbl`, and `data.frame`. The results include all existing columns in `.data` plus new columns for exposures and observation periods. Observation periods include counters for policy exposures, start dates, and end dates. Both start dates and end dates are inclusive bounds.

For policy year exposures, two observation period columns are returned. Columns beginning with (`pol_`) are integer policy periods. Columns beginning with (`pol_date_`) are calendar dates representing anniversary dates, monthiversary dates, etc.

Policy period and calendar period variations

The functions `expose_py()`, `expose_pq()`, `expose_pm()`, `expose_pw()`, `expose_cy()`, `expose_cq()`, `expose_cm()`, `expose_cw()` are convenience functions for specific implementations of `expose()`. The two characters after the underscore describe the exposure type and exposure period, respectively.

For exposures types:

- `p` refers to policy years
- `c` refers to calendar years

For exposure periods:

- `y` = years
- `q` = quarters
- `m` = months
- `w` = weeks

All columns containing dates must be in YYYY-MM-DD format.

References

Atkinson and McGarry (2016). Experience Study Calculations. https://www.soa.org/49378a/globalassets/assets/files/research/experience-study-calculations.pdf

See Also

`expose_split()` for information on splitting calendar year exposures by policy year.

Examples

```
toy_census |> expose("2020-12-31")

census_dat |> expose_py("2019-12-31", target_status = "Surrender")
```

`expose_split` *Split calendar exposures by policy year*

Description

Split calendar period exposures that cross a policy anniversary into a pre-anniversary record and a post-anniversary record.

After splitting the data, the resulting data frame will contain both calendar exposures and policy year exposures. These columns will be named `exposure_cal` and `exposure_pol`, respectively. Calendar exposures will be in the original units passed to `expose_split()`. Policy exposures will always be expressed in years.

After splitting exposures, downstream functions like `exp_stats()` and `exp_shiny()` will require clarification as to which exposure basis should be used to summarize results.

`is_split_exposed_df()` will return `TRUE` if `x` is a `split_exposed_df` object.

Usage

```
expose_split(.data)

is_split_exposed_df(x)
```

Arguments

`.data`	An `exposed_df` object with calendar period exposures.
`x`	Any object

Details

`.data` must be an `exposed_df` with calendar year, quarter, month, or week exposure records. Calendar year exposures are created by the functions `expose_cy()`, `expose_cq()`, `expose_cm()`, or `expose_cw()`, (or `expose()` when `cal_expo = TRUE`).

Value

For `expose_split()`, a tibble with class `split_exposed_df`, `exposed_df`, `tbl_df`, `tbl`, and `data.frame`. The results include all columns in `.data` except that `exposure` has been renamed to `exposure_cal`. Additional columns include:

- `exposure_pol` - policy year exposures
- `pol_yr` - policy year

For `is_split_exposed_df()`, a length-1 logical vector.

See Also

`expose()` for information on creating exposure records from census data.

Examples

```
toy_census |> expose_cy("2022-12-31") |> expose_split()
```

`exp_shiny` *Interactively explore experience data*

Description

Launch a Shiny application to interactively explore drivers of experience.

`dat` must be an `exposed_df` object. An error will be thrown is any other object type is passed. If `dat` has transactions attached, the app will contain features for both termination and transaction studies. Otherwise, the app will only support termination studies.

If nothing is passed to `predictors`, all columns names in `dat` will be used (excluding the policy number, status, termination date, exposure, transaction counts, and transaction amounts columns).

The `expected` argument is optional. As a default, any column names containing the word "expected" are used.

Usage

```
exp_shiny(
  dat,
  predictors = names(dat),
  expected = names(dat)[grepl("expected", names(dat))],
  distinct_max = 25L,
  title,
  credibility = TRUE,
  conf_level = 0.95,
  cred_r = 0.05,
  theme = "shiny",
  col_exposure = "exposure"
)
```

Arguments

`dat`	An `exposed_df` object.
`predictors`	A character vector of independent variables in `dat` to include in the Shiny app.
`expected`	A character vector of expected values in `dat` to include in the Shiny app.
`distinct_max`	Maximum number of distinct values allowed for `predictors` to be included as "Color" and "Facets" grouping variables. This input prevents the drawing of overly complex plots. Default value = 25.

title	Optional. Title of the Shiny app. If no title is provided, a descriptive title will be generated based on attributes of dat.
credibility	If TRUE, the output will include partial credibility weights and credibility-weighted termination rates.
conf_level	Confidence level used for the Limited Fluctuation credibility method and confidence intervals
cred_r	Error tolerance under the Limited Fluctuation credibility method
theme	The name of a theme passed to the preset argument of bslib::bs_theme(). Alternatively, a complete Bootstrap theme created using bslib::bs_theme().
col_exposure	Name of the column in dat containing exposures. This input is only used to clarify the exposure basis when dat is a split_exposed_df object. For more information on split exposures, see expose_split().

Value

No return value. This function is called for the side effect of launching a Shiny application.

Layout

Filters:

The sidebar contains filtering widgets organized by data type for all variables passed to the predictors argument.

At the top of the sidebar, information is shown on the percentage of records remaining after applying filters. A description of all active filters is also provided.

The top of the sidebar also includes a "play / pause" switch that can pause reactivity of the application. Pausing is a good option when multiple changes are made in quick succession, especially when the underlying data set is large.

Grouping variables:

This box includes widgets to select grouping variables for summarizing experience. The "x" widget determines the x variable in the plot output. Similarly, the "Color" and "Facets" widgets are used for color and facets. Multiple faceting variable selections are allowed. For the table output, "x", "Color", and "Facets" have no particular meaning beyond the order in which grouping variables are displayed.

Study type:

This box includes a toggle to switch between termination studies and transaction studies (if available). Different options are available for each study type.

Termination studies:
The expected values checkboxes are used to activate and deactivate expected values passed to the expected argument. These checkboxes also include a a "control" item for expected values derived using control variables. These boxes impact the table output directly and the available "y" variables for the plot. The "Weight by" widget is used to specify which column, if any, contains weights for summarizing experience. The "Control variables" widget is used to specify which columns, if any, are used as control variables (see exp_stats() for more information).

Transaction studies:
The transaction types checkboxes are used to activate and deactivate transaction types that appear in the plot and table outputs. The available transaction types are taken from the `trx_types` attribute of `dat`. In the plot output, transaction type will always appear as a faceting variable. The "Transactions as % of" selector will expand the list of available "y" variables for the plot and impact the table output directly. Lastly, a toggle exists that allows for all transaction types to be aggregated into a single group.

Output:

Plot:
This tab includes a plot and various options for customization:

- y: y variable
- Geometry: plotting geometry
- Second y-axis: activate to enable a second y-axis
- Second axis y: y variable to plot on the second axis
- Add Smoothing: activate to plot loess curves
- Confidence intervals: If available, add error bars for confidence intervals around the selected y variable
- Free y Scales: activate to enable separate y scales in each plot
- Log y-axis: activate to plot all y-axes on a log-10 scale

The gear icon above the plot contains a pop-up menu that can be used to change the size of the plot for exporting.

Table:
This tab includes a data table.
The gear icon above the table contains a pop-up menu that can be used to change the appearance of the table:

- The "Total row", "Confidence intervals", and "Credibility-weighted termination rates" switches add these outputs to the table. These values are hidden as a default to prevent over-crowding.
- The "Include color scales" switch disables or re-enables conditional color formatting.
- The "Decimals" slider controls the number of decimals displayed for percentage fields.
- The "Font size multiple" slider impacts the table's font size

Export:
This pop-up menu contains options for saving summarized experience data, the plot, or the table. Data is saved as a CSV file. The plot and table are saved as png files.

Examples

```
if (interactive()) {
  study_py <- expose_py(census_dat, "2019-12-31", target_status = "Surrender")
  expected_table <- c(seq(0.005, 0.03, length.out = 10),
                      0.2, 0.15, rep(0.05, 3))

  study_py <- study_py |>
    mutate(expected_1 = expected_table[pol_yr],
           expected_2 = ifelse(inc_guar, 0.015, 0.03)) |>
    add_transactions(withdrawals) |>
    left_join(account_vals, by = c("pol_num", "pol_date_yr"))
```

```
  exp_shiny(study_py)
}
```

`exp_stats` *Summarize experience study records*

Description

Create a summary data frame of termination experience for a given target status.

Usage

```
exp_stats(
  .data,
  target_status = attr(.data, "target_status"),
  expected,
  col_exposure = "exposure",
  col_status = "status",
  wt = NULL,
  credibility = FALSE,
  conf_level = 0.95,
  cred_r = 0.05,
  conf_int = FALSE,
  control_vars,
  control_distinct_max = 25L
)

## S3 method for class 'exp_df'
summary(object, ...)
```

Arguments

`.data`	A data frame with exposure-level records, ideally of type `exposed_df`
`target_status`	A character vector of target status values
`expected`	A character vector containing column names in `.data` with expected values
`col_exposure`	Name of the column in `.data` containing exposures
`col_status`	Name of the column in `.data` containing the policy status
`wt`	Optional. Length 1 character vector. Name of the column in `.data` containing weights to use in the calculation of claims, exposures, partial credibility, and confidence intervals.
`credibility`	If `TRUE`, the output will include partial credibility weights and credibility-weighted termination rates.
`conf_level`	Confidence level used for the Limited Fluctuation credibility method and confidence intervals

cred_r	Error tolerance under the Limited Fluctuation credibility method
conf_int	If TRUE, the output will include confidence intervals around the observed termination rates and any actual-to-expected ratios.
control_vars	".none" or a character vector containing column names in .data to use as control variables
control_distinct_max	Maximum number of unique values allowed for control variables
object	An exp_df object
...	Groups to retain after summary() is called

Details

If .data is grouped, the resulting data frame will contain one row per group.

If target_status isn't provided, exp_stats() will use the same target status from .data if it has the class exposed_df. Otherwise, all status values except the first level will be assumed. This will produce a warning message.

Value

A tibble with class exp_df, tbl_df, tbl, and data.frame. The results include columns for any grouping variables, claims, exposures, and observed termination rates (q_obs).

- If any values are passed to expected or control_vars, additional columns are added for expected termination rates and actual-to-expected (A/E) ratios. A/E ratios are prefixed by ae_.
- If credibility is set to TRUE, additional columns are added for partial credibility and credibility-weighted termination rates (assuming values are passed to expected). Credibility-weighted termination rates are prefixed by adj_.
- If conf_int is set to TRUE, additional columns are added for lower and upper confidence interval limits around the observed termination rates and any actual-to-expected ratios. Additionally, if credibility is TRUE and expected values are passed to expected, the output will contain confidence intervals around credibility-weighted termination rates. Confidence interval columns include the name of the original output column suffixed by either _lower or _upper.
- If a value is passed to wt, additional columns are created containing the the sum of weights (.weight), the sum of squared weights (.weight_qs), and the number of records (.weight_n).

Expected values

The expected argument is optional. If provided, this argument must be a character vector with values corresponding to column names in .data containing expected experience. More than one expected basis can be provided.

Control variables

The control_vars argument is optional. If provided, this argument must be ".none" (more on this below) or a character vector with values corresponding to column names in .data. Control variables are used to estimate the impact of any grouping variables on observed experience *after accounting for* the impact of control variables.

Mechanically, when values are passed to control_vars, a separate call is made to exp_stats() using the control variables as grouping variables. This is used to derive a new expected values basis called control, which is both added to .data and appended to the expected argument. In the final output, a column called ae_control shows the relative impact of any grouping variables after accounting for the control variables.

About ".none": If ".none" is passed to control_vars, a single aggregate termination rate is calculated for the entire data set and used to compute control and ae_control.

The control_distinct_max argument places an upper limit on the number of unique values that a control variable is allowed to have. This limit exists to prevent an excessive number of groups on continuous or high-cardinality features.

It should be noted that usage of control variables is a rough approximation and not a substitute for rigorous statistical models. The impact of control variables is calculated in isolation and does consider other features or possible confounding variables. As such, control variables are most useful for exploratory data analysis.

Credibility

If credibility is set to TRUE, the output will contain a credibility column equal to the partial credibility estimate under the Limited Fluctuation credibility method (also known as Classical Credibility) assuming a binomial distribution of claims.

Confidence intervals

If conf_int is set to TRUE, the output will contain lower and upper confidence interval limits for the observed termination rate and any actual-to-expected ratios. The confidence level is dictated by conf_level. If no weighting variable is passed to wt, confidence intervals will be constructed assuming a binomial distribution of claims. Otherwise, confidence intervals will be calculated assuming that the aggregate claims distribution is normal with a mean equal to observed claims and a variance equal to:

Var(S) = E(N) * Var(X) + E(X)^2 * Var(N),

Where S is the aggregate claim random variable, X is the weighting variable assumed to follow a normal distribution, and N is a binomial random variable for the number of claims.

If credibility is TRUE and expected values are passed to expected, the output will also contain confidence intervals for any credibility-weighted termination rates.

summary() Method

Applying summary() to a exp_df object will re-summarize the data while retaining any grouping variables passed to the "dots" (...).

References

Herzog, Thomas (1999). Introduction to Credibility Theory

Examples

```
toy_census |> expose("2022-12-31", target_status = "Surrender") |>
    exp_stats()

exp_res <- census_dat |>
           expose("2019-12-31", target_status = "Surrender") |>
           group_by(pol_yr, inc_guar) |>
           exp_stats(control_vars = "product")

exp_res
summary(exp_res)
summary(exp_res, inc_guar)
```

`is_exposed_df` *Exposed data frame helper functions*

Description

Test for and coerce to the `exposed_df` class.

Usage

```
is_exposed_df(x)

as_exposed_df(
  x,
  end_date,
  start_date = as.Date("1900-01-01"),
  target_status = NULL,
  cal_expo = FALSE,
  expo_length = c("year", "quarter", "month", "week"),
  trx_types = NULL,
  col_pol_num,
  col_status,
  col_exposure,
  col_pol_per,
  cols_dates,
  col_trx_n_ = "trx_n_",
  col_trx_amt_ = "trx_amt_",
  default_status
)
```

Arguments

x	An object. For as_exposed_df(), x must be a data frame.
end_date	Experience study end date
start_date	Experience study start date. Default value = 1900-01-01.
target_status	Character vector of target status values. Default value = NULL.
cal_expo	Set to TRUE for calendar year exposures. Otherwise policy year exposures are assumed.
expo_length	Exposure period length
trx_types	Optional. Character vector containing unique transaction types that have been attached to x. For each value in trx_types, as_exposed_df requires that columns exist in x named trx_n_{*} and trx_amt_{*} containing transaction counts and amounts, respectively. The prefixes "trx_n_" and "trx_amt_" can be overridden using the col_trx_n_ and col_trx_amt_ arguments.
col_pol_num	Optional. Name of the column in x containing the policy number. The assumed default is "pol_num".
col_status	Optional. Name of the column in x containing the policy status. The assumed default is "status".
col_exposure	Optional. Name of the column in x containing exposures. The assumed default is "exposure".
col_pol_per	Optional. Name of the column in x containing policy exposure periods. Only necessary if cal_expo is FALSE. The assumed default is either "pol_yr", "pol_qtr", "pol_mth", or "pol_wk" depending on the value of expo_length.
cols_dates	Optional. Names of the columns in x containing exposure start and end dates. Both date ranges are assumed to be exclusive. The assumed default is of the form *A_B*. *A* is "cal" if cal_expo is TRUE or "pol" otherwise. *B* is either "yr", "qtr", "mth", or "wk" depending on the value of expo_length.
col_trx_n_	Optional. Prefix to use for columns containing transaction counts.
col_trx_amt_	Optional. Prefix to use for columns containing transaction amounts.
default_status	Optional scalar character representing the default active status code. If not provided, the most common status is assumed.

Details

is_exposed_df() will return TRUE if x is an exposed_df object.

as_exposed_df() will coerce a data frame to an exposed_df object if that data frame has columns for policy numbers, statuses, exposures, policy periods (for policy exposures only), and exposure start / end dates. Optionally, if x has transaction counts and amounts by type, these can be specified without calling add_transactions().

Value

For is_exposed_df(), a length-1 logical vector. For as_exposed_df(), an exposed_df object.

See Also

expose() for information on how exposed_df objects are typically created from census data.

---

plot_special *Additional plotting functions for termination studies*

Description

These functions create additional experience study plots that are not available or difficult to produce using the autoplot.exp_df() function.

Usage

```
plot_termination_rates(object, ..., include_cred_adj = FALSE)

plot_actual_to_expected(object, ..., add_hline = TRUE)
```

Arguments

object	An object of class exp_df created by the function exp_stats().
...	Additional arguments passed to autoplot.exp_df().
include_cred_adj	If TRUE, credibility-weighted termination rates will be plotted as well.
add_hline	If TRUE, a blue dashed horizontal line will be drawn at 100%.

Details

plot_termination_rates() - Create a plot of observed termination rates and any expected termination rates attached to an exp_df object.

plot_actual_to_expected() - Create a plot of actual-to-expected termination rates attached to an exp_df object.

Value

a ggplot object

See Also

autoplot.exp_df()

Examples

```
study_py <- expose_py(census_dat, "2019-12-31", target_status = "Surrender")
expected_table <- c(seq(0.005, 0.03, length.out = 10), 0.2, 0.15, rep(0.05, 3))

study_py <- study_py |>
  mutate(expected_1 = expected_table[pol_yr],
         expected_2 = ifelse(inc_guar, 0.015, 0.03))

exp_res <- study_py |> group_by(pol_yr) |>
  exp_stats(expected = c("expected_1", "expected_2"))

plot_termination_rates(exp_res)

plot_actual_to_expected(exp_res)
```

---

plot_special_trx *Additional plotting functions for transaction studies*

Description

These functions create additional experience study plots that are not available or difficult to produce using the `autoplot.trx_df()` function.

Usage

```
plot_utilization_rates(object, ...)
```

Arguments

`object`	An object of class `trx_df` created by the function `trx_stats()`.
`...`	Additional arguments passed to `autoplot.trx_df()`.

Details

`plot_utilization_rates()` - Create a plot of transaction frequency and severity. Frequency is represented by utilization rates (`trx_util`). Severity is represented by transaction amounts as a percentage of one or more other columns in the data (`{*}_w_trx`). All severity series begin with the prefix "pct_of_" and end with the suffix "_w_trx". The suffix refers to the fact that the denominator only includes records with non-zero transactions. Severity series are based on column names passed to the `percent_of` argument in `trx_stats()`. If no "percentage of" columns exist in `object`, this function will only plot utilization rates.

Value

a `ggplot` object

See Also

autoplot.trx_df()

Examples

```
study_py <- expose_py(census_dat, "2019-12-31",
                      target_status = "Surrender") |>
  add_transactions(withdrawals) |>
  left_join(account_vals, by = c("pol_num", "pol_date_yr"))

trx_res <- study_py |> group_by(pol_yr) |>
  trx_stats(percent_of = "av_anniv", combine_trx = TRUE)

plot_utilization_rates(trx_res)
```

---

pol_yr *Calculate policy duration*

Description

Given a vector of dates and a vector of issue dates, calculate policy years, quarters, months, or weeks.

Usage

```
pol_yr(x, issue_date)

pol_qtr(x, issue_date)

pol_mth(x, issue_date)

pol_wk(x, issue_date)
```

Arguments

x	A vector of dates
issue_date	A vector of issue dates

Details

These functions assume the first day of each policy year is the anniversary date (or issue date in the first year). The last day of each policy year is the day before the next anniversary date. Analogous rules are used for policy quarters, policy months, and policy weeks.

Value

An integer vector

Examples

```
pol_yr(as.Date("2021-02-28") + 0:2, "2020-02-29")

pol_mth(as.Date("2021-02-28") + 0:2, "2020-02-29")
```

---

qx_iamb *2012 Individual Annuity Mortality Table and Projection Scale G2*

Description

Mortality rates and mortality improvement rates from the 2012 Individual Annuity Mortality Basic (IAMB) Table and Projection Scale G2.

Usage

```
qx_iamb

scale_g2
```

Format

For the 2012 IAMB table, a data frame with 242 rows and 3 columns:

age Attained age

qx Mortality rate

gender Female or Male

For the Projection Scale G2 table, a data frame with 242 rows and 3 columns:

age Attained age

mi Mortality improvement rate

gender Female or Male

Source

- https://mort.soa.org/
- https://www.actuary.org/sites/default/files/files/publications/Payout_Annuity_Report_09-28-11.pdf

`sim_data` *Simulated annuity data*

Description

Simulated data for a theoretical deferred annuity product with an optional guaranteed income rider. This data is theoretical only and does not represent the experience on any specific product.

Usage

```
census_dat

withdrawals

account_vals
```

Format

Three data frames containing census records (`census_dat`), withdrawal transactions (`withdrawals`), and historical account values (`account_vals`).

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 20000 rows and 11 columns.

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 160130 rows and 4 columns.

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 141252 rows and 3 columns.

Census data (`census_dat`)

pol_num Policy number

status Policy status: Active, Surrender, or Death

issue_date Issue date

inc_guar Indicates whether the policy was issued with an income guarantee

qual Indicates whether the policy was purchased with tax-qualified funds

age Issue age

product Product: a, b, or c

gender M (Male) or F (Female)

wd_age Age that withdrawals commence

premium Single premium deposit

term_date Termination date upon death or surrender

Withdrawal data (`withdrawals`)

pol_num Policy number

trx_date Withdrawal transaction date

trx_type Withdrawal transaction type, either Base or Rider

trx_amt Withdrawal transaction amount

Account values data (`account_vals`)

pol_num Policy number

pol_date_yr Policy anniversary date (beginning of year)

av_anniv Account value on the policy anniversary date

See Also

census_dat

`step_expose` *Create exposure records in a* `recipes` *step*

Description

`step_expose()` creates a *specification* of a recipe step that will convert a data frame of census-level records to exposure-level records.

Usage

```
step_expose(
  recipe,
  ...,
  role = NA,
  trained = FALSE,
  end_date,
  start_date = as.Date("1900-01-01"),
  target_status = NULL,
  options = list(cal_expo = FALSE, expo_length = "year"),
  drop_pol_num = TRUE,
  skip = TRUE,
  id = recipes::rand_id("expose")
)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose variables for this step. See `selections()` for more details.
`role`	Not used by this step since no new variables are created.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.
`end_date`	Experience study end date
`start_date`	Experience study start date. Default value = 1900-01-01.
`target_status`	Character vector of target status values. Default value = `NULL`.

options	A named list of additional arguments passed to expose().
drop_pol_num	Whether the pol_num column produced by expose() should be dropped. Defaults to TRUE.
skip	A logical. Should the step be skipped when the recipe is baked by bake()? While all operations are baked when prep() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations.
id	A character string that is unique to this step to identify it.

Details

Policy year exposures are calculated as a default. To switch to calendar exposures or another exposure length, use pass the appropriate arguments to the options parameter.

Policy numbers are dropped as a default whenever the recipe is baked. This is done to prevent unintentional errors when the model formula includes all variables (y ~ .). If policy numbers are required for any reason (mixed effect models, identification, etc.), set drop_pol_num to FALSE.

Value

An updated version of recipe with the new expose step added to the sequence of any existing operations. For the tidy method, a tibble with the columns exposure_type, target_status, start_date, and end_date.

See Also

expose()

Examples

```
expo_rec <- recipes::recipe(status ~ ., toy_census) |>
  step_expose(end_date = "2022-12-31", target_status = "Surrender",
              options = list(expo_length = "month")) |>
  prep()

recipes::juice(expo_rec)
```

---

summary.exposed_df *Summarize experience study records*

Description

Create a summary data frame of termination experience for a given target status.

Usage

```
## S3 method for class 'exposed_df'
summary(object, ...)
```

Arguments

object	A data frame with exposure-level records
...	Additional arguments passed to exp_stats()

Details

Calling summary() on an exposed_df object will summarize results using exp_stats(). See exp_stats() for more information.

Value

A tibble with class exp_df, tbl_df, tbl, and data.frame.

See Also

exp_stats()

Examples

```
toy_census |> expose("2022-12-31", target_status = "Surrender") |>
    summary()
```

---

toy_census *Toy policy census data*

Description

A tiny dataset containing 3 policies: one active, one terminated due to death, and one terminated due to surrender.

Usage

```
toy_census
```

Format

A data frame with 3 rows and 4 columns:

pol_num Policy number

status Policy status

issue_date Issue date

term_date Termination date

trx_stats *Summarize transactions and utilization rates*

Description

Create a summary data frame of transaction counts, amounts, and utilization rates.

Usage

```
trx_stats(
  .data,
  trx_types,
  percent_of = NULL,
  combine_trx = FALSE,
  col_exposure = "exposure",
  full_exposures_only = TRUE,
  conf_int = FALSE,
  conf_level = 0.95
)

## S3 method for class 'trx_df'
summary(object, ...)
```

Arguments

.data	A data frame with exposure-level records of type exposed_df with transaction data attached. If necessary, use as_exposed_df() to convert a data frame to an exposed_df object, and use add_transactions() to attach transactions to an exposed_df object.
trx_types	A character vector of transaction types to include in the output. If none is provided, all available transaction types in .data will be used.
percent_of	A optional character vector containing column names in .data to use as denominators in the calculation of utilization rates or actual-to-expected ratios.
combine_trx	If FALSE (default), the results will contain output rows for each transaction type. If TRUE, the results will contains aggregated experience across all transaction types.
col_exposure	Name of the column in .data containing exposures
full_exposures_only	If TRUE (default), partially exposed records will be excluded from data.
conf_int	If TRUE, the output will include confidence intervals around the observed utilization rate and any percent_of output columns.
conf_level	Confidence level for confidence intervals
object	A trx_df object
...	Groups to retain after summary() is called

Details

Unlike exp_stats(), this function requires data to be an exposed_df object.

If .data is grouped, the resulting data frame will contain one row per transaction type per group.

Any number of transaction types can be passed to the trx_types argument, however each transaction type **must** appear in the trx_types attribute of .data. In addition, trx_stats() expects to see columns named trx_n_{*} (for transaction counts) and trx_amt_{*} for (transaction amounts) for each transaction type. To ensure .data is in the appropriate format, use the functions as_exposed_df() to convert an existing data frame with transactions or add_transactions() to attach transactions to an existing exposed_df object.

Value

A tibble with class trx_df, tbl_df, tbl, and data.frame. The results include columns for any grouping variables and transaction types, plus the following:

- trx_n: the number of unique transactions.
- trx_amt: total transaction amount
- trx_flag: the number of observation periods with non-zero transaction amounts.
- exposure: total exposures
- avg_trx: mean transaction amount (trx_amt / trx_flag)
- avg_all: mean transaction amount over all records (trx_amt / exposure)
- trx_freq: transaction frequency when a transaction occurs (trx_n / trx_flag)
- trx_util: transaction utilization per observation period (trx_flag / exposure)

If percent_of is provided, the results will also include:

- The sum of any columns passed to percent_of with non-zero transactions. These columns include the suffix _w_trx.
- The sum of any columns passed to percent_of
- pct_of_{*}_w_trx: total transactions as a percentage of column {*}_w_trx. In other words, total transactions divided by the sum of a column including only records utilizing transactions.
- pct_of_{*}_all: total transactions as a percentage of column {*}. In other words, total transactions divided by the sum of a column regardless of whether or not transactions were utilized.

If conf_int is set to TRUE, additional columns are added for lower and upper confidence interval limits around the observed utilization rate and any percent_of output columns. Confidence interval columns include the name of the original output column suffixed by either _lower or _upper.

- If values are passed to percent_of, an additional column is created containing the the sum of squared transaction amounts (trx_amt_sq).

"Percentage of" calculations

The `percent_of` argument is optional. If provided, this argument must be a character vector with values corresponding to columns in `.data` containing values to use as denominators in the calculation of utilization rates or actual-to-expected ratios. Example usage:

- In a study of partial withdrawal transactions, if `percent_of` refers to account values, observed withdrawal rates can be determined.
- In a study of recurring claims, if `percent_of` refers to a column containing a maximum benefit amount, utilization rates can be determined.

Confidence intervals

If `conf_int` is set to `TRUE`, the output will contain lower and upper confidence interval limits for the observed utilization rate and any `percent_of` output columns. The confidence level is dictated by `conf_level`.

- Intervals for the utilization rate (`trx_util`) assume a binomial distribution.
- Intervals for transactions as a percentage of another column with non-zero transactions (`pct_of_{*}_w_trx`) are constructed using a normal distribution
- Intervals for transactions as a percentage of another column regardless of transaction utilization (`pct_of_{*}_all`) are calculated assuming that the aggregate distribution is normal with a mean equal to observed transactions and a variance equal to:

  `Var(S) = E(N) * Var(X) + E(X)^2 * Var(N)`,

  Where `S` is the aggregate transactions random variable, `X` is an individual transaction amount assumed to follow a normal distribution, and `N` is a binomial random variable for transaction utilization.

Default removal of partial exposures

As a default, partial exposures are removed from `.data` before summarizing results. This is done to avoid complexity associated with a lopsided skew in the timing of transactions. For example, if transactions can occur on a monthly basis or annually at the beginning of each policy year, partial exposures may not be appropriate. If a policy had an exposure of 0.5 years and was taking withdrawals annually at the beginning of the year, an argument could be made that the exposure should instead be 1 complete year. If the same policy was expected to take withdrawals 9 months into the year, it's not clear if the exposure should be 0.5 years or 0.5 / 0.75 years. To override this treatment, set `full_exposures_only` to `FALSE`.

`summary()` Method

Applying `summary()` to a `trx_df` object will re-summarize the data while retaining any grouping variables passed to the "dots" (`...`).

Examples

```
expo <- expose_py(census_dat, "2019-12-31", target_status = "Surrender") |>
  add_transactions(withdrawals)

res <- expo |> group_by(inc_guar) |> trx_stats(percent_of = "premium")
```

```
res

summary(res)

expo |> group_by(inc_guar) |>
  trx_stats(percent_of = "premium", combine_trx = TRUE, conf_int = TRUE)
```

Index

∗ **datasets**
agg_sim_dat, 4
qx_iamb, 30
sim_data, 31
toy_census, 34

account_vals *(*sim_data*)*, 31
add_predictions, 2
add_transactions, 3
add_transactions(), *26*, *35, 36*
agg_sim_dat, 4
as_exp_df, 5
as_exposed_df *(*is_exposed_df*)*, 25
as_exposed_df(), *4*, *35, 36*
as_trx_df, 8
autoplot(), *7*, *9*
autoplot.exp_df *(*autoplot_exp*)*, 11
autoplot.exp_df(), *27*
autoplot.trx_df *(*autoplot_exp*)*, 11
autoplot.trx_df(), *28, 29*
autoplot_exp, 11
autotable, 13
autotable(), *7*, *9*

bake(), *33*

census_dat, *5*, *32*
census_dat *(*sim_data*)*, 31

exp_shiny, 19
exp_stats, 22
exp_stats(), *7*, *12*, *14*, *20*, *23, 24*, *27*, *34*, *36*
expose, 15
expose(), *4*, *18, 19*, *27*, *33*
expose_cm *(*expose*)*, 15
expose_cm(), *18*
expose_cq *(*expose*)*, 15
expose_cq(), *18*
expose_cw *(*expose*)*, 15
expose_cw(), *18*
expose_cy *(*expose*)*, 15
expose_cy(), *18*
expose_pm *(*expose*)*, 15
expose_pq *(*expose*)*, 15
expose_pw *(*expose*)*, 15
expose_py *(*expose*)*, 15
expose_split, 18
expose_split(), *17*, *20*

ggplot2::facet_wrap(), *12*
ggplot2::ggplot(), *12*
ggplot2::scale_y_continuous(), *12*
gt::cols_label(), *14*
gt::fmt_number(), *14*
gt::gt(), *14*

is_exp_df *(*as_exp_df*)*, 5
is_exposed_df, 25
is_split_exposed_df *(*expose_split*)*, 18
is_trx_df *(*as_trx_df*)*, 8

paletteer::palettes_d_names, *15*
plot_actual_to_expected *(*plot_special*)*, *27*
plot_actual_to_expected(), *12*, *27*
plot_special, 27
plot_special_trx, 28
plot_termination_rates *(*plot_special*)*, *27*
plot_termination_rates(), *12*, *27*
plot_utilization_rates *(*plot_special_trx*)*, 28
plot_utilization_rates(), *28*
pol_mth *(*pol_yr*)*, 29
pol_qtr *(*pol_yr*)*, 29
pol_wk *(*pol_yr*)*, 29
pol_yr, 29
predict(), *3*
prep(), *33*

qx_iamb, 30

scale_g2 *(*qx_iamb*)*, 30
selections(), *32*
sim_data, 31
step_expose, 32
summary(), *6*, *7*, *9*, *34*
summary.exp_df *(*exp_stats*)*, 22
summary.exposed_df, 33
summary.trx_df *(*trx_stats*)*, 35

toy_census, 34
trx_stats, 35
trx_stats(), *10*, *12*, *14*, *28*

withdrawals *(*sim_data*)*, 31