Package 'actxps'

March 4, 2023

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

Version 1.0.0

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, 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 7.2.3

Suggests knitr, RColorBrewer, rmarkdown, testthat (>= 3.0.0), shiny (>= 1.6), bslib, thematic

Config/testthat/edition 3

Depends R (>= 4.1)

Imports lubridate, dplyr (>= 1.1.0), ggplot2, tibble, rlang, glue, purrr, scales, gt, paletteer, recipes, generics, readr, tidyr, vctrs

LazyData true

VignetteBuilder knitr

NeedsCompilation no

Author Matt Heaphy [aut, cre]

Repository CRAN

Date/Publication 2023-03-04 14:10:02 UTC

R topics documented:

`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.

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)
```

---

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,
  scales = "fixed",
  geoms = c("lines", "bars"),
  y_labels = scales::label_percent(accuracy = 0.1)
)

## S3 method for class 'trx_df'
autoplot(
  object,
  ...,
  x = NULL,
  y = NULL,
  color = NULL,
  mapping,
  scales = "fixed",
  geoms = c("lines", "bars"),
  y_labels = scales::label_percent(accuracy = 0.1)
)
```

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.
scales	The scales argument passed to ggplot2::facet_wrap().
geoms	Type of geometry. If "points" is passed, the plot will display lines and points. If "bars", the plot will display bars.
y_labels	Label function passed to ggplot2::scale_y_continuous().

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

`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(...),
  ...
)

## 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(...),
  ...
)
```

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 decrement rate and actual-to-expected columns.
color_q_obs	Color palette used for the observed decrement 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.
color_util	Color palette used for utilization rates.
color_pct_of	Color palette used for "percentage of" columns.

Details

See paletteer::paletteer_d()'s palette argument for usage of the color_q_obs and color_ae_ arguments.

Value

a gt object

---

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
...	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. If left blank, then the first status level will be assumed to be the default active status.

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.

References

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

Examples

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

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

`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
)
```

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.

Value

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

Layout

Filters:

The sidebar contains filtering widgets for all variables passed to the `predictors` argument.

Study options:

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

Study type:
This box also includes a toggle to switch between termination studies and transaction studies (if available).

Termination studies:
The expected values checkboxes are used to activate and deactivate expected values passed to the `expected` argument. This impacts the table output directly and the available "y" variables for the plot. If there are no expected values available, this widget will not appear. The "Weight by" widget is used to specify which column, if any, contains weights for summarizing experience.

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 checkbox exists that allows for all transaction types to be aggregated into a single group.

Output:

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

- y: y variable
- Geometry: plotting geometry
- Add Smoothing?: activate to plot loess curves
- Free y Scales: activate to enable separate y scales in each plot.

Table:
This tab includes a data table.

Export Data:
This tab includes a download button that will save a copy of the summarized experience data.

Filter Information:

This box contains information on the original number of exposure records, the number of records after filters are applied, and the percentage of records retained.

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,
  cred_p = 0.95,
  cred_r = 0.05
)

## 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, and partial credibility.
credibility	whether the output should include partial credibility weights and credibility-weighted decrement rates.
cred_p	confidence level under the Limited Fluctuation credibility method
cred_r	error tolerance under the Limited Fluctuation credibility method
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, .data is not an exposed_df object, 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 decrement rates (q_obs). If any values are passed to expected, additional columns will be added for expected decrements and actual-to-expected ratios. If credibility is set to TRUE, additional columns are added for partial credibility and credibility-weighted decrement rates (assuming values are passed to expected).

Expected values

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

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.

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("2020-12-31", target_status = "Surrender") |>
    exp_stats()

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

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_"
)
```

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 "pol_yr", "pol_qtr", "pol_mth", or "pol_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 amount.

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.

---

pol_yr *Calculate policy duration*

Description

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

Usage

```
pol_yr(x, issue_date)

pol_qtr(x, issue_date)

pol_mth(x, issue_date)

pol_wk(x, issue_date)

pol_interval(x, issue_date, dur_length)
```

Arguments

x	A vector of dates
issue_date	A vector of issue dates
dur_length	A period object. See lubridate::period().

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.

The pol_interval() function can be used to determine any arbitrary duration passed to the dur_length argument.

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 Project 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 Project 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

`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)
```

---

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
)

## 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 results 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.
object	an 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_utilization`: 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`
- `pct_of_{*}_all`: total transactions as a percentage of column `{*}`

"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.

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)
```

Index