**R topics documented:**

- autoplot.exp_df ........................................ 2
- autotable ................................................. 3
- census_dat ................................................ 4
- expose ....................................................... 5
- exp_shiny .................................................. 7
- exp_stats ................................................... 9
- is_exposed_df ............................................. 11
- qx_iamb .................................................... 12
- stepExpose .................................................. 12
- toy_census .................................................. 14

**Index**

- autoplot.exp_df ........................................ 2
- Plot experience study results 

---

**Description**

Plot experience study results

**Usage**

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

**Arguments**

- `object` An object of class `exp_df` usually created by the function `exp_stats()`.
- `...` Faceting variables passed to `facet_wrap()`.
- `mapping` Aesthetic mapping passed to `ggplot()`.
- `scales` The `scales` argument passed to `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 `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 all grouping variables 3+ as facets passed into `facet_wrap()`.

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.

Usage

```r
autotable(object, ...)
```

## S3 method for class 'exp_df'
```r
autotable(
  object,
  fontsize = 100,
  decimals = 1,
  colorful = TRUE,
  color_q_obs = "RColorBrewer::GnBu",
  color_ae_ = "RColorBrewer::RdBu",
  rename_cols = rlang::list2(...),
  ...
)
```

Arguments

- `object` An object of class `exp_df` usually created by the function `exp_stats()`.
- `...` 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.
census_dat

color_ae_  Color palette used for actual-to-expected rates.
renamed_cols  An optional list consisting of key-value pairs. This can be used to relabel
columns on the output table. Names are column names in object and values
are new labels. See \texttt{gt::cols_label()} for more information.

Details

See \texttt{paletteer::paletteer_d()}’s \texttt{palette} argument for usage of the \texttt{color\_q\_obs} and \texttt{color\_ae\_}
arguments.

Value

a \texttt{gt} object

census_dat \hspace{1cm} Simulated census data

Description

Simulated census 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

Format

A data frame with 20,000 rows and 11 columns:

\begin{itemize}
  \item \texttt{pol\_num}  policy number
  \item \texttt{status}  policy status: Active, Surrender, or Death
  \item \texttt{issue\_date}  issue date
  \item \texttt{inc\_guar}  indicates whether the policy was issued with an income guarantee
  \item \texttt{qual}  indicates whether the policy was purchased with tax-qualified funds
  \item \texttt{age}  issue age
  \item \texttt{product}  product: a, b, or c
  \item \texttt{gender}  M (Male) or F (Female)
  \item \texttt{wd\_age}  Age that withdrawals commence
  \item \texttt{term\_date}  termination date upon death or surrender
\end{itemize}
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
)

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 used.

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.

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**


**Examples**

```r
toy_census |> expose("2020-12-31")
census_dat |> expose_py("2019-12-31", target_status = "Surrender")
```

---

**Description**

Launch a shiny application to interactively explore drivers of experience.

dat must be an exposed_df object. An error will be thrown if any other object type is passed.

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

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

**Usage**

```r
exp_shiny(
  dat,
  predictors = names(dat),
  expected = stringr::str_subset(names(dat), "expected"),
  distinct_max = 25L
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dat</td>
<td>An exposed_df object.</td>
</tr>
<tr>
<td>predictors</td>
<td>A character vector of independent variables in dat to include in the shiny app.</td>
</tr>
<tr>
<td>expected</td>
<td>A character vector of expected values in dat to include in the shiny app.</td>
</tr>
<tr>
<td>distinct_max</td>
<td>Maximum number of distinct values allowed for predictors to be included as &quot;Color&quot; and &quot;Facets&quot; grouping variables. This input prevents the drawing of overly complex plots. Default value = 25.</td>
</tr>
</tbody>
</table>
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.

Variable Selection:
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.

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 in 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.

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

```r
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))
  set.seed(123)
  study_py <- study_py |
  dplyr::mutate(expected_1 = expected_table[pol_yr],
                expected_2 = ifelse(inc_guar, 0.015, 0.03),
                weights = rnorm(nrow(study_py), 100, 50) |> abs())
}
exp_stats

exp_shiny(study_py)
}

---

exp_stats  Summarize experience study records

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

Usage

```r
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
)
```

```r
## 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") |
  dplyr::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 = "year"
)

Arguments

x an object. x must be a data frame when calling as_exposed_df()
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 used.
expo_length exposure period length

Details

These are behind-the-scenes functions that will generally not be called by users.
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.

Value

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

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

```r
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

```r
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
Index

* datasets
  census_dat, 4
  qx_iamb, 12
  toy_census, 14

as_exposed_df(is_exposed_df), 11
autoplot.exp_df, 2
autotable, 3

bake(), 13

census_dat, 4

exp_shiny, 7
exp_stats, 9
expose, 5
expose(), 14
expose_cm(expose), 5
expose_cq(expose), 5
expose_cw(expose), 5
expose_cy(expose), 5
expose_pm(expose), 5
expose_pq(expose), 5
expose_pw(expose), 5
expose_py(expose), 5

is_exposed_df, 11

prep(), 13

qx_iamb, 12

scale_g2(qx_iamb), 12
selections(), 13
step_expose, 12
summary.exp_df(exp_stats), 9

toy_census, 14