Package ‘statar’

August 19, 2023

Title  Tools Inspired by 'Stata' to Manipulate Tabular Data
Version  0.7.6
Description  A set of tools inspired by 'Stata' to explore data.frames ('summarize',
             'tabulate', 'xtile', 'pctile', 'binscatter', elapsed quarters/month, lead/lag).
License  GPL-2
URL  https://github.com/matthieugomez/statar
BugReports  https://github.com/matthieugomez/statar/issues
Depends  R (>= 3.2.0)
Imports  data.table, tidyselect, dplyr (>= 1.0), ggplot2 (>= 2.0.0),
           lazyeval, matrixStats, methods, rlang, stringr
Suggests  knitr, lubridate, rmarkdown, testthat
VignetteBuilder  knitr
Encoding  UTF-8
RoxygenNote  7.2.3
NeedsCompilation  no
Author  Matthieu Gomez [aut, cre]
Maintainer  Matthieu Gomez <mg3901@columbia.edu>
Repository  CRAN
Date/Publication  2023-08-19 13:50:02 UTC

R topics documented:

  elapsed ............................................................... 2
  fill_gap ............................................................ 3
  is.panel ........................................................... 4
  join ................................................................. 5
  n_narm ............................................................. 6
  pctile ............................................................. 7
  statar .............................................................. 7
  stat_binmean ....................................................... 7
Elapse dates (monthly, quarterly)

Usage

\begin{verbatim}
as.quarterly(x)
is.quarterly(x)
as.monthly(x)
is.monthly(x)
\end{verbatim}

Argument

\texttt{x} \hspace{1cm} a vector

Details

Monthly and quarterly dates are stored as integers, representing the number of elapsed calendar periods since 01/01/1970. As \texttt{yearmonth} and \texttt{yearqtr} the package \texttt{zoo}, these dates are printed in a way that fits their frequency (\texttt{YYYq, YYYmm}). The only difference is that, monthly, and quarterly are integers, which removes issues due to floating points (particularly important when merging). This also allows to use arithmetic on perios, \texttt{ie date + 1 adds one period rather than one day}.

Methods to convert from and to Dates or POSIXlt are provided. In particular, you may use lubridate \texttt{week month} and \texttt{year} to extract information from elapsed dates.

Examples

\begin{verbatim}
library(lubridate)
library(dplyr)
date <- mdy(c("04/03/1992", "01/04/1992", "03/15/1992"))
datem <- as.monthly(date)
is.monthly(datem)
as.quarterly(date)
as.character(datem)
\end{verbatim}
```r
datem + 1
df <- tibble(datem)
# filter(df, month(datem) == 1)
seq(datem[1], datem[2])
as.Date(datem)
as.POSIXlt(datem)
as.POSIXct(datem)
week(datem)

fill_gap

Add rows corresponding to gaps in some variable

Description
Add rows corresponding to gaps in some variable

Usage
```r
fill_gap(
  x,
  ..., 
  full = FALSE,
  roll = FALSE,
  rollends = if (roll == "nearest") c(TRUE, TRUE) else if (roll >= 0) c(FALSE, TRUE) else 
    c(TRUE, FALSE)
)
```r

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>A data frame</td>
</tr>
<tr>
<td>...</td>
<td>a time variable</td>
</tr>
<tr>
<td>full</td>
<td>A boolean. When full = FALSE (default), the function creates rows corresponding to all missing times between the min and max of ... within each group. When full = TRUE, the function creates rows corresponding to all missing times between the min and max of ... in the whole dataset.</td>
</tr>
<tr>
<td>roll</td>
<td>When roll is a positive number, values are carried forward. roll=TRUE is equivalent to roll=+Inf. When roll is a negative number, values are rolled backwards; i.e., next observation carried backwards (NOCB). Use -Inf for unlimited roll back. When roll is &quot;nearest&quot;, the nearest value is used. Default to FALSE (no rolling)</td>
</tr>
<tr>
<td>rollends</td>
<td>A logical vector length 2 (a single logical is recycled). When rolling forward (e.g. roll=TRUE) if a value is past the last observation within each group defined by the join columns, rollends[2]=TRUE will roll the last value forwards. rollends[1]=TRUE will roll the first value backwards if the value is before it. If rollends=FALSE the value of i must fall in a gap in x but not after the end or before the beginning of the data, for that group defined by all but the last join column. When roll is a finite number, that limit is also applied when rolling the end</td>
</tr>
</tbody>
</table>
Examples

```r
library(dplyr)
library(lubridate)

df <- tibble(
  id = c(1, 1, 1, 1),
  datem = as.monthly(mdy(c("01/01/1992", "02/01/1992", "04/01/1992", "7/11/1992"))),
  value = c(4.1, 4.5, 3.3, 3.2)
)

df %>% group_by(id) %>% fill_gap(datem)

df %>% group_by(id) %>% fill_gap(datem, roll = 1)

df %>% group_by(id) %>% fill_gap(datem, roll = "nearest")

df %>% group_by(id) %>% fill_gap(datem, roll = "nearest", full = TRUE)
```

is.panel

Check whether a data.frame is a panel

Description

Check whether a data.frame is a panel

Usage

```r
is.panel(x, ...)
```

Arguments

- `x`: a data frame
- `...`: a time variable

Value

The function `is.panel` check that there are no duplicate combinations of the variables in `...` and that no observation is missing for the last variable in `...` (the time variable).

Examples

```r
library(dplyr)

df <- tibble(
  id1 = c(1, 1, 1, 2, 2),
  id2 = 1:5,
  value = c(4.1, 4.5, 3.3, 3.2, 5.2)
)

df %>% group_by(id1) %>% is.panel(year)

df1 <- df %>% filter(!is.na(year))

df1 %>% is.panel(year)

df1 %>% group_by(id1) %>% is.panel(year)

df1 %>% group_by(id1, id2) %>% is.panel(year)
```
join

Join two data frames together

Description

Join two data frames together

Usage

```r
join(
  x,
  y,
  kind,
  on = intersect(names(x), names(y)),
  suffixes = c("\ .x", "\ .y"),
  check = m ~ m,
  gen = FALSE,
  inplace = FALSE,
  update = FALSE,
  type
)
```

Arguments

- **x**: The master data.frame
- **y**: The using data.frame
- **kind**: The kind of (SQL) join among "full" (default), "left", "right", "inner", "semi", "anti" and "cross".
- **on**: Character vectors specifying variables to match on. Default to common names between x and y.
- **suffixes**: A character vector of length 2 specifying suffix of overlapping columns. Default to ".x" and ".y".
- **check**: A formula checking for the presence of duplicates. Specifying 1~m (resp m~1, 1~1) checks that joined variables uniquely identify observations in x (resp y, both).
- **gen**: Name of new variable to mark result, or the boolean FALSE (default) if no such variable should be created. The variable equals 1 for rows in master only, 2 for rows in using only, 3 for matched rows.
- **inplace**: A boolean. In case "kind"= "left" and RHS of check is 1, the merge can be one in-place.
- **update**: A boolean. For common variables in x and y not specified in "on", replace missing observations by the non missing observations in y.
- **type**: Deprecated
Value

A data.frame that joins rows in master and using datasets. Importantly, if x or y are not keyed, the join may change their row orders.

Examples

library(dplyr)
x <- data.frame(a = rep(1:2, each = 3), b=1:6)
y <- data.frame(a = 0:1, bb = 10:11)
join(x, y, kind = "full")
join(x, y, kind = "left", gen = ".merge")
join(x, y, kind = "right", gen = ".merge")
join(x, y, kind = "inner", check = m~1)
join(x, y, kind = "semi")
join(x, y, kind = "anti")
y <- rename(y, b = bb)
join(x, y, kind = "full", on = "a")
join(x, y, kind = "full", on = "a", suffixes = c("",".i"))
y <- data.frame(a = 0:1, bb = 10:11)
join(x, y, kind = "left", check = m~1)
x <- data.frame(a = c(1,2), b=c(NA, 2))
y <- data.frame(a = c(1,2), b = 10:11)
join(x, y, kind = "left", on = "a", update = TRUE)
join(x, y, kind = "left", on = "a", check = m~1, update = TRUE)

Description

Count number of non missing observations

Usage

n_narm(...)

Arguments

... a sequence of vectors, matrices and data frames.

Examples

n_narm(1:100, c(NA, 1:99))
pctile

Weighted quantile of type 2 (similar to Stata _pctile)

Description

Weighted quantile of type 2 (similar to Stata _pctile)

Usage

pctile(x, probs = c(0.25, 0.5, 0.75), wt = NULL, na.rm = FALSE)

Arguments

x A vector
probs A vector of probabilities
wt A weight vector
na.rm Should missing values be returned?

statar

A package for applied research

Description

A package for applied research

stat_binmean

Plot the mean of y over the mean of x within bins of x.

Description

Plot the mean of y over the mean of x within bins of x.

Usage

stat_binmean(  
  mapping = NULL,  
  data = NULL,  
  geom = "point",  
  position = "identity",  
  show.legend = NA,  
  inherit.aes = TRUE,  
  na.rm = FALSE,  
  n = 20,  
  ...  
)

}
stat_binmean

Arguments

mapping Set of aesthetic mappings created by \texttt{aes()}. If specified and \texttt{inherit.aes} = \texttt{TRUE} (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.

data The data to be displayed in this layer. There are three options:
If \texttt{NULL}, the default, the data is inherited from the plot data as specified in the call to \texttt{ggplot()}. A \texttt{data.frame}, or other object, will override the plot data. All objects will be fortified to produce a data frame. See \texttt{fortify()} for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a \texttt{data.frame}, and will be used as the layer data. A function can be created from a formula (e.g. \texttt{~ head(x, 10)}).

geom The geometric object to use to display the data, either as a ggproto Geom subclass or as a string naming the geom stripped of the geom_ prefix (e.g. "point" rather than "geom_point")

position Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use \texttt{position_jitter}), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.

show.legend logical. Should this layer be included in the legends? \texttt{NA}, the default, includes if any aesthetics are mapped. \texttt{FALSE} never includes, and \texttt{TRUE} always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes If \texttt{FALSE}, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. \texttt{borders()}.

na.rm If \texttt{FALSE} (the default), removes missing values with a warning. If \texttt{TRUE} silently removes missing values.

n number of x-bins. Default to 20. Set to zero if you want to use distinct value of x for grouping.

... Other arguments passed on to \texttt{layer()}. These are often aesthetics, used to set an aesthetic to a fixed value, like \texttt{colour = "red"} or \texttt{size = 3}. They may also be parameters to the paired geom/stat.

Value

a \texttt{data.frame} with additional columns:

<table>
<thead>
<tr>
<th>xtile</th>
<th>bins for x</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>mean of x</td>
</tr>
<tr>
<td>y</td>
<td>mean of y</td>
</tr>
</tbody>
</table>

Examples

library(ggplot2)
g <- ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length)) + stat_binmean(n = 10)
sum_up

\[ g + \text{stat_smooth}(\text{method = "lm", se = FALSE}) \]

\[
\text{ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Species)) + stat_binmean(n = 10)} \\
\text{ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, weight = Petal.Length)) + stat_binmean(n = 10)} \\
\]

---

**sum_up**  
*Gives summary statistics (corresponds to Stata command *summarize*)*

**Description**

Gives summary statistics (corresponds to Stata command *summarize*).

**Usage**

```
sum_up(df, ..., d = FALSE, wt = NULL)
```

**Arguments**

- `df`: a data.frame
- `...`: Variables to include. Defaults to all non-grouping variables. See the `select` documentation.
- `d`: Should detailed summary statistics be printed?
- `wt`: Weights. Default to NULL.

**Value**

a data.frame

**Examples**

```r
library(dplyr)
N <- 100
df <- tibble(
  id = 1:N,
  v1 = sample(5, N, TRUE),
  v2 = sample(1e6, N, TRUE)
)
sum_up(df)
sum_up(df, v2, d = TRUE)
sum_up(df, v2, wt = v1)
df %>% group_by(v1) %>% sum_up(starts_with("v"))
```
Description

Returns cross tabulation

Usage

\[
\text{tab}(x, \ldots, wt = \text{NULL}, \text{na.rm} = \text{FALSE}, \text{sort} = \text{TRUE})
\]

Arguments

- **x**: a vector or a data.frame
- **\ldots**: Variable(s) to include. If length is two, a special cross tabulation table is printed although a long data.frame is always (invisibly) returned.
- **wt**: Frequency weights. Default to NULL.
- **na.rm**: Remove missing values. Default to FALSE
- **sort**: Boolean. Default to TRUE

Value

a data.frame sorted by variables in \ldots, and with columns "Freq.", "Percent", and "Cum." for counts.

Examples

```r
# setup
c\texttt{library(dplyr)}
N <- 1e2 \; \text{; \texttt{K = 10}}
df <- \texttt{tibble(}
  \texttt{id = sample(c(NA,1:5), N/K, TRUE),}
  \texttt{v1 = sample(1:5, N/K, TRUE)}
\texttt{)}
\texttt{# one-way tabulation}
df \texttt{\%\% tab(id)}
df \texttt{\%\% tab(id, wt = v1)}
\texttt{# two-way tabulation}
df \texttt{\%\% tab(id, v1)}
df \texttt{\%\% filter(id \geq 3) \%\% tab(id)}
```
**tempname**

Create unique names within a list, a data.frame, or an environment

**Description**

Create unique names within a list, a data.frame, or an environment

**Usage**

`tempname(where = globalenv(), n = 1, prefix = ".temp", inherits = TRUE)`

**Arguments**

- `where`: A character vector, list or an environment
- `n`: An integer that specifies length of the output
- `prefix`: A character vector that specifies prefix for new name
- `inherits`: Should the name unique also in the enclosing frames of the environment?

**Examples**

- `tempname(c("temp1", "temp3"), 4)`
- `tempname(globalenv())`
- `tempname(data.frame(temp = 1), n = 3)`

---

**tlead-tlag**

lead and lag with respect to a time variable

**Description**

lead and lag with respect to a time variable

**Usage**

- `tlead(x, n = 1L, time, default = NA)`
- `tlag(x, n = 1L, time, default = NA)`

**Arguments**

- `x`: a vector of values
- `n`: a positive integer of length 1, giving the number of positions to lead or lag by. When the package lubridate is loaded, it can be a period when using with time (see the lubridate function minutes, hours, days, weeks, months and years)
- `time`: time variable
- `default`: value used for non-existant rows. Defaults to NA.
Examples

value <- c(4.1, 4.5, 3.3)
tlag(value, 1, time = date) # returns value in year - 1
library(lubridate)
date <- as.monthly(mdy(c("01/04/1992", "03/15/1992", "04/03/1992")))
tlag(value, time = date)
library(dplyr)
df <- tibble(
id = c(1, 2, 2),
date = date,
value = value
)
df %>% group_by(id) %>% mutate(valuel = tlag(value, n = 1, time = date))

---

winsorize

Winsorize a numeric vector

Description

Winsorize a numeric vector

Usage

winsorize(
  x,
  probs = NULL,
  cutpoints = NULL,
  replace = c(cutpoints[1], cutpoints[2]),
  verbose = TRUE
)

winsorise(
  x,
  probs = NULL,
  cutpoints = NULL,
  replace = c(cutpoints[1], cutpoints[2]),
  verbose = TRUE
)

Arguments

x A vector of values

probs A vector of probabilities that can be used instead of cutpoints. Quantiles are computed as the inverse of the empirical distribution function (type = 1)
cutpoints Cutpoints under and above which are defined outliers. Default is (median - five times interquartile range, median + five times interquartile range). Compared to bottom and top percentile, this takes into account the whole distribution of the vector.

replace Values by which outliers are replaced. Default to cutpoints. A frequent alternative is NA.

verbose Boolean. Should the percentage of replaced values printed?

Examples

v <- c(1:4, 99)
winsorize(v)
winsorize(v, replace = NA)
winsorize(v, probs = c(0.01, 0.99))
winsorize(v, cutpoints = c(1, 50))

xtile Bin variable in groups (similar to Stata xtile)

Description

Bin variable in groups (similar to Stata xtile)

Usage

xtile(x, n = NULL, probs = NULL, cutpoints = NULL, wt = NULL)

Arguments

x A vector
n A numeric specifying number of quantiles. Can be used instead of cutpoints
probs A vector of probabilities that an be used instead of cutpoints. Quantiles are computed as the inverse of the empirical distribution function (type = 1)
cutpoints Cutpoints to use when nq is not specified. For instance cutpoints = 0.4 creates two groups, one for observations equal or below 0.4, one for observations superior to 0.4.
wt A variable specifying weight in case the option n_quantiles is specified.

Value

An integer vector representing groups corresponding to cutpoints. Includes missing values when present in the original vector.
Examples

\[ x <- \text{c(NA, 1:10)} \]
\[ \text{xtile}(x, n = 3) \] # 3 groups based on terciles
\[ \text{xtile}(x, \text{probs} = \text{c(0.3, 0.7)}) \] # 3 groups based on two quantiles
\[ \text{xtile}(x, \text{cutpoints} = \text{c(2, 3)}) \] # 3 groups based on two cutpoints
Index

aes(), 8
as.monthly (elapsed), 2
as.quarterly (elapsed), 2
borders(), 8
elapsed, 2
fill_gap, 3
fortify(), 8
ggplot(), 8
is.monthly (elapsed), 2
is.panel, 4
is.quarterly (elapsed), 2
join, 5
layer(), 8
month, 2
monthly (elapsed), 2
n_narm, 6
pctile, 7
quarterly, (elapsed), 2
select, 9
stat_binmean, 7
statar, 7
statar-package (statar), 7
sum_up, 9
tab, 10
tempname, 11
tlag(tlead-tlag), 11
tlead(tlead-tlag), 11
tlead-tlag, 11
week, 2
winsorise (winsorize), 12
winsorize, 12
xtitle, 13
year, 2