Package: tbrf (via r-universe)

October 17, 2026

Type Package

Title Time-Based Rolling Functions

Version 0.1.7

Description Provides rolling statistical functions based on date and time windows instead of n-lagged observations.

URL https://mps9506.github.io/tbrf/

BugReports https://github.com/mps9506/tbrf/issues

License GPL-3 | file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

Depends R (>= 2.10), ggplot2 (>= 2.2.1)

Imports boot, dplyr, lubridate, purrr, rlang, stats, tibble, tidyr

Suggests spelling, covr, testthat, knitr, rmarkdown

VignetteBuilder knitr

Language en-US

Config/Needs/website mps9506/mpsTemplates

Config/pak/sysreqs libicu-dev

Repository https://mps9506.r-universe.dev

Date/Publication 2025-08-19 13:32:30 UTC

RemoteUrl https://github.com/mps9506/tbrf

RemoteRef HEAD

RemoteSha 8fa0344d1c93a540e04672fd4d23f82a3f0228cb

Contents

Dissolved_Oxygen 2
Entero 3
stat_stepribbon 3
tbr_binom 5
tbr_gmean 6
tbr_mean 7
tbr_median 8
tbr_misc 9
tbr_sd 10
tbr_sum 11
tbrf 12
tbrf-ggproto 12

Index **13**

`Dissolved_Oxygen` *Dissolved oxygen measurements from the Tres Palacios River*

Description

Data from the Texas Commission on Environmental Quality Surface Water Quality Monitoring Information System. The 'AverageDO" field is the mean of dissolved oxygen concentrations (mg/L) measured at a field site at that day. The MinDO is the minimum dissolved oxygen concentration measured at that site on that day.

Usage

```
data(Dissolved_Oxygen)
```

Format

A data frame with 236 rows and 6 variables:

Station_ID unique water quality monitoring station identifier

Date sampling date in yyyy-mm-dd format

Param_Code unique parameter code

Param_Desc parameter description with units

Average_DO mean of dissolved oxygen measurement, in mg/L

Min_DO minimum of dissolved oxygen measurement, in mg/L

Source

https://www80.tceq.texas.gov/SwqmisPublic/public/default.htm

Entero *Enterococci bacteria measurements from the Tres Palacios River*

Description

Data from the Texas Commission on Environmental Quality Surface Water Quality Monitoring Information System. The ‘Value“ field is the lab measured value of Enterococci bacteria (MPN/100 mL) from grab samples collected at ‘Station ID‘ on the Tres Palacios River on ‘Date‘.

Usage

```
data(Entero)
```

Format

A data frame with 212 rows and 5 variables:

Station_ID unique water quality monitoring station identifier

Date sampling date in yyyy-mm-dd format

Param_Code unique parameter code

Param_Desc parameter description with units

Value Enterococci concentration, in MPN/L

Source

https://www80.tceq.texas.gov/SwqmisPublic/public/default.htm

stat_stepribbon *Step ribbon statistic*

Description

Provides stairstep values for ribbon plots. This was originally in Bob Rudis’s ggalt package which is no longer on CRAN.

Usage

```
stat_stepribbon(
  mapping = NULL,
  data = NULL,
  geom = "ribbon",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
```

```
  direction = "hv",
  ...
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = 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 NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See 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 data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
geom	which geom to use; defaults to "ribbon"
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If 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. borders().
direction	hv for horizontal-veritcal steps, vh for vertical-horizontal steps
...	Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth = 3. The geom's documentation has an **Aesthetics** section that lists the

available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.

- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

Author(s)

Bob Rudis

References

https://groups.google.com/forum/?fromgroups=#!topic/ggplot2/9cFWHaH1CPs

Examples

```
x <- 1:10
df <- data.frame(x=x, y=x+10, ymin=x+7, ymax=x+12)

gg <- ggplot(df, aes(x, y))
gg <- gg + geom_ribbon(aes(ymin=ymin, ymax=ymax),
                       stat="stepribbon", fill="#b2b2b2")
gg <- gg + geom_step(color="#2b2b2b")
gg

gg <- ggplot(df, aes(x, y))
gg <- gg + geom_ribbon(aes(ymin=ymin, ymax=ymax),
                       stat="stepribbon", fill="#b2b2b2",
                       direction="hv")
gg <- gg + geom_step(color="#2b2b2b")
gg
```

---

tbr_binom *Time-Based Rolling Binomial Probability*

Description

Produces a a rolling time-window based vector of binomial probability and confidence intervals.

Usage

```
tbr_binom(.tbl, x, tcolumn, unit = "years", n, alpha = 0.05, na.pad = TRUE)
```

Arguments

.tbl	dataframe with two variables.
x	indicates the variable column containing "success" and "failure" observations coded as 1 or 0.
tcolumn	indicates the variable column containing Date or Date-Time values.
unit	character, one of "years", "months", "weeks", "days", "hours", "minutes", "seconds"
n	numeric, describing the length of the time window in the selected units.
alpha	numeric, probability of a type 1 error, so confidence coefficient = 1-alpha
na.pad	logical. If 'na.pad = TRUE' incomplete windows (duration of the window < 'n') return 'NA'. Defatuls to 'TRUE'

Value

tibble with binomial point estimate and confidence intervals.

See Also

binom_ci

Examples

```
## Generate Sample Data
df <- tibble::tibble(
date = sample(seq(as.Date('2000-01-01'), as.Date('2015/12/30'), by = "day"), 100),
value = rbinom(100, 1, 0.25)
)

## Run Function
tbr_binom(df, x = value,
tcolumn = date, unit = "years", n = 5,
alpha = 0.1, na.pad = FALSE)
```

---

tbr_gmean *Time-Based Rolling Geometric Mean*

Description

Produces a a rolling time-window based vector of geometric means and confidence intervals.

Usage

```
tbr_gmean(.tbl, x, tcolumn, unit = "years", n, na.pad = TRUE, ...)
```

Arguments

.tbl	a data frame with at least two variables; time column formatted as date, date/time and value column.
x	column containing the values to calculate the geometric mean.
tcolumn	formatted time column.
unit	character, one of "years", "months", "weeks", "days", "hours", "minutes", "seconds"
n	numeric, describing the length of the time window.
na.pad	logical. If 'na.pad = TRUE' incomplete windows (duration of the window < 'n') return 'NA'. Defatuls to 'TRUE'
...	additional arguments passed to gm_mean_ci

Value

tibble with columns for the rolling geometric mean and upper and lower confidence levels.

See Also

gm_mean_ci

Examples

```
## Return a tibble with new rolling geometric mean column
tbr_gmean(Dissolved_Oxygen, x = Average_DO, tcolumn = Date, unit = "years", n = 5, na.pad = FALSE)

## Not run:
## Return a tibble with rolling geometric mean and 95% CI
tbr_gmean(Dissolved_Oxygen, x = Average_DO, tcolumn = Date, unit = "years", n = 5, conf = .95)
## End(Not run)
```

---

tbr_mean *Time-Based Rolling Mean*

Description

Produces a a rolling time-window based vector of means and confidence intervals.

Usage

```
tbr_mean(.tbl, x, tcolumn, unit = "years", n, na.pad = TRUE, ...)
```

Arguments

.tbl	a data frame with at least two variables; time column formatted as date, date/time and value column.
x	column containing the numeric values to calculate the mean.
tcolumn	formatted time column.
unit	character, one of "years", "months", "weeks", "days", "hours", "minutes", "seconds"
n	numeric, describing the length of the time window.
na.pad	logical. If 'na.pad = TRUE' incomplete windows (duration of the window < 'n') return 'NA'. Defaults to 'TRUE'
...	additional arguments passed to mean_ci.

Value

tibble with columns for the rolling mean and upper and lower confidence intervals.

See Also

mean_ci

Examples

```
## Return a tibble with new rolling mean column
tbr_mean(Dissolved_Oxygen, x = Average_DO, tcolumn = Date, unit = "years", n = 5, na.pad = FALSE)

## Not run:
## Return a tibble with rolling mean and 95% CI
tbr_mean(Dissolved_Oxygen, x = Average_DO, tcolumn = Date, unit = "years", n = 5, conf = .95)
## End(Not run)
```

---

tbr_median *Time-Based Rolling Median*

Description

Produces a a rolling time-window based vector of medians and confidence intervals.

Usage

```
tbr_median(.tbl, x, tcolumn, unit = "years", n, na.pad = TRUE, ...)
```

Arguments

.tbl	a data frame with at least two variables; time column formatted as date, date/time and value column.
x	column containing the numeric values to calculate the mean.
tcolumn	formatted time column.
unit	character, one of "years", "months", "weeks", "days", "hours", "minutes", "seconds"
n	numeric, describing the length of the time window.
na.pad	logical. If 'na.pad = TRUE' incomplete windows (duration of the window < 'n') return 'NA'. Defaults to 'TRUE'
...	additional arguments passed to median_ci

Value

tibble with columns for the rolling median and upper and lower confidence intervals.

See Also

median_ci

Examples

```
## Return a tibble with new rolling median column
tbr_median(Dissolved_Oxygen, x = Average_DO, tcolumn = Date, unit = "years",
n = 5, na.pad = FALSE)

## Not run:
## Return a tibble with rolling median and 95% CI
tbr_median(Dissolved_Oxygen, x = Average_DO, tcolumn = Date, unit = "years", n = 5, conf = .95)
## End(Not run)
```

---

tbr_misc *Use Generic Functions with Time Windows*

Description

Use Generic Functions with Time Windows

Usage

```
tbr_misc(.tbl, x, tcolumn, unit = "years", n, na.pad = TRUE, func, ...)
```

Arguments

.tbl	a data frame with at least two variables; time column formatted as date, date/time and value column.
x	column containing the values the function is applied to.
tcolumn	formatted time column.
unit	character, one of "years", "months", "weeks", "days", "hours", "minutes", "seconds"
n	numeric, describing the length of the time window.
na.pad	logical. If 'na.pad = TRUE' incomplete windows (duration of the window < 'n') return 'NA'. Defaults to 'TRUE'
func	specified function
...	optional additional arguments passed to function func

Value

tibble

Examples

```
tbr_misc(Dissolved_Oxygen, x = Average_DO, tcolumn = Date, unit = "years",
n = 5, na.pad = FALSE, func = mean)
```

---

tbr_sd *Time-Based Rolling Standard Deviation*

Description

Time-Based Rolling Standard Deviation

Usage

```
tbr_sd(.tbl, x, tcolumn, unit = "years", n, na.rm = FALSE, na.pad = TRUE)
```

Arguments

.tbl	a data frame with at least two variables; time column formatted as date, date/time and value column.
x	column containing the values to calculate the standard deviation.
tcolumn	formatted time column.
unit	character, one of "years", "months", "weeks", "days", "hours", "minutes", "seconds"
n	numeric, describing the length of the time window.
na.rm	logical. Should missing values be removed?
na.pad	logical. If 'na.pad = TRUE' incomplete windows (duration of the window < 'n') return 'NA'. Defaults to 'TRUE'

Value

tibble with column for the rolling sd.

See Also

sd

Examples

```
tbr_sd(Dissolved_Oxygen, x = Average_DO, tcolumn = Date, unit = "years", n = 5, na.pad = FALSE)
```

---

tbr_sum *Time-Based Rolling Sum*

Description

Time-Based Rolling Sum

Usage

```
tbr_sum(.tbl, x, tcolumn, unit = "years", n, na.rm = FALSE, na.pad = TRUE)
```

Arguments

.tbl	a data frame with at least two variables; time column formatted as date, date/time and value column.
x	column containing the values to calculate the sum.
tcolumn	formatted time column.
unit	character, one of "years", "months", "weeks", "days", "hours", "minutes", "seconds"
n	numeric, describing the length of the time window.
na.rm	logical. Should missing values be removed?
na.pad	logical. If 'na.pad = TRUE' incomplete windows (duration of the window < 'n') return 'NA'. Defatuls to 'TRUE'

Value

dataframe with column for the rolling sum.

See Also

sum

Examples

```
tbr_sum(Dissolved_Oxygen, x = Average_DO, tcolumn = Date, unit = "years", n =
5, na.pad = FALSE)
```

tbrf *Time-Based Rolling Functions*

Description

Provides rolling statistical functions based on date and time windows instead of n-lagged observations.

Author(s)

Michael Schramm

See Also

Useful links:

- https://mps9506.github.io/tbrf/
- Report bugs at https://github.com/mps9506/tbrf/issues

tbrf-ggproto *tbrf extensions to ggplot2*

Description

tbrf makes use of the ggproto class system to extend the functionality of ggplot2. In general the actual classes should be of little interest to users as the standard ggplot2 api of using geom_* and stat_* functions for building up the plot is encouraged.

References

https://groups.google.com/forum/?fromgroups=#!topic/ggplot2/9cFWHaH1CPs

Index

∗ **datasets**
    Dissolved_Oxygen, 2
    Entero, 3
    tbrf-ggproto, 12

aes(), *4*

binom_ci, *6*
borders(), *4*

Dissolved_Oxygen, 2

Entero, 3

fortify(), *4*

ggplot(), *4*
gm_mean_ci, *7*

key glyphs, *5*

layer position, *4*
layer(), *4*, *5*

mean_ci, *8*
median_ci, *9*

sd, *11*
stat_stepribbon, 3
StatStepribbon *(*tbrf-ggproto*)*, 12
sum, *11*

tbr_binom, 5
tbr_gmean, 6
tbr_mean, 7
tbr_median, 8
tbr_misc, 9
tbr_sd, 10
tbr_sum, 11
tbrf, 12
tbrf-ggproto, 12
tbrf-package *(*tbrf*)*, 12