Package ‘tsbox’

August 6, 2019

Type Package

Title Class-Agnostic Time Series

Version 0.2.0

Date 2019-08-06

Description Time series toolkit with identical behavior for all time series classes: 'ts','xts', 'data.frame', 'data.table', 'tibble', 'zoo', 'timeSeries', 'tsibble', 'tis' or 'irts'. Also converts reliably between these classes.

Imports data.table (>= 1.10.0), anytime

Suggests testthat, dplyr, tibble, forecast, seasonal, dygraphs, xts, ggplot2, scales, knitr, rmarkdown, tsibble (>= 0.8.2), tsibbledata, tibbletime, tseries, zoo, tis, timeSeries, nycflights13

License GPL-3

Encoding UTF-8

URL https://www.tsbox.help

BugReports https://github.com/christophsax/tsbox/issues

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Christoph Sax [aut, cre] (<https://orcid.org/0000-0002-7192-7044>)

Maintainer Christoph Sax <christoph.sax@gmail.com>

Repository CRAN

Date/Publication 2019-08-06 06:40:02 UTC

R topics documented:

  tsbox-package .................................................. 2
  copy_class ..................................................... 3
  relevant_class .................................................. 3
tsbox-package

tsbox: Class-Agnostic Time Series

Description

The R ecosystem knows a vast number of time series classes: ts, xts, zoo, tsibble, tibbletime, tis, or timeSeries. The plethora of standards causes confusion. As different packages rely on different classes, it is hard to use them in the same analysis. tsbox provides a set of tools that make it easy to switch between these classes. It also allows the user to treat time series as plain data frames, facilitating the use with tools that assume rectangular data.

Details

The package is built around a set of functions that convert time series of different classes to each other. They are frequency-agnostic, and allow the user to combine multiple non-standard and irregular frequencies. Because coercion works reliably, it is easy to write functions that work identically for all classes. So whether we want to smooth, scale, differentiate, chain-link, forecast, regularize or seasonally adjust a time series, we can use the same tsbox-command for any time series class.

The best way to start is to check out the package website.
**copy_class**

**Author(s)**
Christoph Sax <christoph.sax@gmail.com>

---

**copy_class**

*Re-Class ts-Boxable Object*

**Description**
Copies class attributes from an existing ts-boxable series. Mainly used internally.

**Usage**
```
copy_class(x, template, preserve.mode = TRUE, preserve.names = FALSE, 
            preserve.time = FALSE)
```

**Arguments**
- **x**: ts-boxable time series, an object of class `ts`, `xts`, `zoo`, `data.frame`, `data.table`, `tbl`, `tbl_ts`, `tbl_time`, `tis`, `irts` or `timeSeries`.
- **template**: ts-boxable time series, an object of class `ts`, `xts`, `data.frame`, or `tibble`. Template.
- **preserve.mode**: should the mode the time column be preserved (data frame only)
- **preserve.names**: should the name of the time column be preserved (data frame only)
- **preserve.time**: should the values time column be preserved (data frame only)

**Details**
Inspired by `xts::reclass`, which does something similar.

---

**relevant_class**

*Extract Relevant Class*

**Description**
Mainly used internally.

**Usage**
```
relevant_class(x)
```

**Arguments**
- **x**: ts-boxable time series, an object of class `ts`, `xts`, `zoo`, `data.frame`, `data.table`, `tbl`, `tbl_ts`, `tbl_time`, `tis`, `irts` or `timeSeries`. 
Examples

relevant_class(AirPassengers)
relevant_class(ts_df(AirPassengers))

---

tsbox-deprecated

**Start and end of time series**

**Description**

Start and end of time series

**Usage**

```r
    ts_start(x)
    ts_end(x)
```

**Arguments**

- `x`: ts-boxable time series, an object of class `ts`, `xts`, `zoo`, `data.frame`, `data.table`, `tbl`, `tbl_ts`, `tbl_time`, `tis`, `irts` or `timeSeries`.

---

**Constructing ts-Functions**

**Description**

`ts_` turns an existing function into a function that can deal with ts-boxable time series objects.

**Usage**

```r
    load_suggested(pkg)
    ts_(fun, class = "ts", vectorize = FALSE, reclass = TRUE)
    ts_apply(x, fun, ...)
```

**Arguments**

- `pkg`: external package, to be suggested (automatically added by `ts_`) `predict()`.
  (See examples)
- `fun`: function, to be made available to all time series classes
- `class`: class that the function uses as its first argument
- `vectorize`: should the function be vectorized? (not yet implemented)
reclass: logical, should the new function return the same same ts-boxable output as imputed?

x: ts-boxable time series, an object of class ts, xts, zoo, data.frame, data.table, tbl, tbl_ts, tbl_time, tis, irts or timeSeries.

... arguments passed to subfunction

Details

The ts_ function is a constructor function for tsbox time series functions. It can be used to wrap any function that works with time series. The default is set to R base "ts" class. ts_ deals with the conversion stuff, 'vectorizes' the function so that it can be used with multiple time series.

Value

A function that accepts ts-boxable time series as an input.

See Also

ts_examples, for a few useful examples of functions generated by ts_.

Vignette on how to make arbitrary functions ts-boxable.

Examples

ts_(rowSums)(ts_c(mdeaths, fdeaths))
ts_plot(mean = ts_(rowMeans)(ts_c(mdeaths, fdeaths)), mdeaths, fdeaths)
ts_(function(x) predict(prcomp(x)))(ts_c(mdeaths, fdeaths))
ts_(function(x) predict(prcomp(x, scale = TRUE))(ts_c(mdeaths, fdeaths))
ts_(dygraphs::dygraph, class = "xts")

# attach series to search path
ts_attach <- ts_(attach, class = "tslist", reclass = FALSE)
ts_attach(EuStockMarkets)
ts_plot(DAX, SMI)
detach()
Usage

e1 %ts+% e2

e1 %ts-% e2

e1 %ts*% e2

e1 %ts/% e2

Arguments

e1 ts-boxable time series, an object of class ts, xts, data.frame, data.table, or tibble.

e2 ts-boxable time series, an object of class ts, xts, data.frame, data.table, or tibble.

Value

a ts-boxable time series, with the same class as the left input.

Examples

head(fdeaths - mdeaths)
head(fdeaths %ts-% mdeaths)
head(ts_df(fdeaths) %ts-% mdeaths)

---

**ts_bind**

*Bind Time Series*

Description

Combine time series to a new, single time series. `ts_bind` combines time series as they are, `ts_chain` chains them together, using percentage change rates.

Usage

ts_bind(...)

ts_chain(...)
Value

A ts-boxable object of the same class as the input. If series of different classes are combined, the class of the first series is used (if possible).

See Also

ts_c to collect multiple time series

Examples

ts_bind(ts_span(mdeaths, end = "1975-12-01"), fdeaths)
ts_bind(mdeaths, c(2, 2))
ts_bind(mdeaths, 3, ts_bind(fdeaths, c(99, 2)))
ts_bind(ts_dt(mdeaths), AirPassengers)

# numeric vectors
ts_bind(12, AirPassengers, c(2, 3))
ts_chain(ts_span(mdeaths, end = "1975-12-01"), fdeaths)

ts_plot(ts_pc(ts_c(
    comb = ts_chain(ts_span(mdeaths, end = "1975-12-01"), fdeaths),
    fdeaths
)))
Collect Time Series

Description

Collect time series as multiple time series.

Usage

\[ \text{ts}_c(...) \]

Arguments

\[ \ldots \]

... ts-boxable time series, objects of class \text{ts}, \text{xts}, \text{data.frame}, \text{data.table}, or \text{tibble}.

Details

In data frame objects, multiple time series are stored in a long data frame. In \text{ts} and \text{xts} objects, time series are combined horizontally.

Value

A ts-boxable object of the same class as the input. If series of different classes are combined, the class of the first series is used (if possible).

See Also

\text{ts_bind}, to bind multiple time series to a single series.

Examples

\begin{verbatim}
head(ts_c(ts_df(EuStockMarkets), AirPassengers))

# labeling
x <- ts_c(
    'International Airline Passengers' = ts_xts(AirPassengers),
    'Deaths from Lung Diseases' = ldeaths
)
head(x)
\end{verbatim}
### ts_default

**Default Column Names**

In data frame objects (data.frame, tibble, data.table), tsbox automatically detects the time and the value column. This function changes the column names to the defaults (time, value), so that auto-detection can be avoided in future operations.

**Usage**

```r
ts_default(x)
```

**Arguments**

- **x**: ts-boxable time series, an object of class ts, xts, zoo, data.frame, data.table, tbl, tbl_ts, tbl_time, tis, irts or timeSeries.

**Value**

a ts-boxable time series, with the same class as the input.

**Examples**

```r
df <- ts_df(ts_c(mdeaths, fdeaths))
# non-default colnames
colnames(df) <- c("id", "date", "count")
# switch back to default colnames
head(ts_default(df))
```

### ts_dts

**Internal Time Series Class**

**Description**

Internal Time Series Class

**Usage**

```r
ts_dts(x)
```

**Arguments**

- **x**: ts-boxable time series, an object of class ts, xts, zoo, data.frame, data.table, tbl, tbl_ts, tbl_time, tis, irts or timeSeries.
Description

Example Functions, Generated by `ts`. `ts_prcomp` calculates the principal components of multiple time series, `ts_dygraphs` generates an interactive graphical visualization, `ts_forecast` return an univariate forecast, `ts_seas` the seasonally adjusted series.

Usage

```r
ts_prcomp(x, ...)  
ts_dygraphs(x, ...)  
ts_forecast(x, ...)  
ts_seas(x, ...)
```

Arguments

- `x`: ts-boxable time series, an object of class `ts`, `xts`, `zoo`, `data.frame`, `data.table`, `tbl`, `tbl_ts`, `tbl_time`, `tis`, `irts` or `timeSeries`.
- `...`: further arguments, passed to the underlying function. For help, consider these functions, e.g., `stats::prcomp`.

Details

With the exception of `ts_prcomp`, these functions depend on external packages.

Value

Usually, a ts-boxable time series, with the same class as the input. `ts_dygraphs` draws a plot.

See Also

Vignette on how to make arbitrary functions ts-boxable.

Examples

```r
ts_plot(  
  ts_scale(ts_c(  
    Male = mdeaths,
    Female = fdeaths,
    'First principal component' = -ts_prcomp(ts_c(mdeaths, fdeaths))[1]
  )  
),  
title = "Deaths from lung diseases",
```
ts_frequency

subtitle = "Normalized values"
)
ts_plot(ts_c(
male = mdeaths, female = fdeaths,
ts_forecast(ts_c('male (fct)' = mdeaths, 'female (fct)' = fdeaths)),
title = "Deaths from lung diseases",
subtitle = "Exponential smoothing forecast"
)
ts_plot(
'Raw series' = AirPassengers,
'Adjusted series' = ts_seas(AirPassengers),
title = "Airline passengers",
subtitle = "X-13 seasonal adjustment"
)

ts_dygraphs(ts_c(mdeaths, EuStockMarkets))

<table>
<thead>
<tr>
<th>ts_frequency</th>
<th>Change Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Changes the frequency of a time series. By default, incomplete periods of regular series are omitted.</td>
</tr>
<tr>
<td>Usage</td>
<td>ts_frequency(x, to = &quot;year&quot;, aggregate = &quot;mean&quot;, na.rm = FALSE)</td>
</tr>
</tbody>
</table>
| Arguments    | x ts-boxable time series, an object of class ts, xts, zoo, data.frame, data.table, tbl, tbl_ts, tbl_time, tis, irt, tis or timeSeries.
to desired frequency, either a character string ("year", "quarter", "month") or an integer (1, 4, 12).
aggregate character string, or function. Either "mean", "sum", "first", or "last", or any aggregate function, such as base::mean().
na.rm logical, if TRUE, incomplete periods are aggregated as well. For irregular series, incomplete periods are always aggregated.

Value

a ts-boxable time series, with the same class as the input.
Examples

```r
ts_frequency(cbind(mdeaths, fdeaths), "year", "sum")
ts_frequency(cbind(mdeaths, fdeaths), "quarter", "last")
ts_frequency(AirPassengers, 4, "sum")

# Note that incomplete years are omitted by default
ts_frequency(EuStockMarkets, "year")
ts_frequency(EuStockMarkets, "year", na.rm = TRUE)
```

---

### ts_ggplot

**Plot Time Series, Using ggplot2**

#### Description

`ts_ggplot()` has the same syntax and produces a similar plot as `ts_plot()`, but uses the `ggplot2` graphic system, and can be customized. With `theme_tsbox()` and `scale_color_tsbox()`, the output of `ts_ggplot` has a similar look and feel.

#### Usage

```r
ts_ggplot(..., title, subtitle, ylab = "")

theme_tsbox(base_family = getOption("ts_font", ""), base_size = 12)

colors_tsbox()

scale_color_tsbox(...)

scale_fill_tsbox(...)
```

#### Arguments

- `...`: ts-boxable time series, objects of class `ts`, `xts`, `data.frame`, `data.table`, or `tibble`. For `scale_` functions, arguments passed to subfunctions.
- `title`: title (optional)
- `subtitle`: subtitle (optional)
- `ylab`: ylab (optional)
- `base_family`: base font family (can also be set via options)
- `base_size`: base font size

#### Details

Both `ts_plot()` and `ts_ggplot()` combine multiple ID dimensions into a single dimension. To plot multiple dimensions in different shapes, facets, etc., use standard ggplot (see examples).
See Also

`ts_plot()`, for a simpler and faster plotting function. `ts_dygraphs()`, for interactive time series plots.

Examples

```r
# using the ggplot2 graphic system
p <- ts_ggplot(total = ldeaths, female = fdeaths, male = mdeaths)
p

# with themes for the look and feel of ts_plot()
p + theme_tsbox() + scale_color_tsbox()

# also use themes with standard ggplot
suppressMessages(library(ggplot2))
df <- ts_df(ts_c(total = ldeaths, female = fdeaths, male = mdeaths))
ggplot(df, aes(x = time, y = value)) +
  facet_wrap("id") +
  geom_line() +
  theme_tsbox() +
  scale_color_tsbox()

## Not run:
library(dataseries)
dta <- ds(c("GDP.PBRTT.A.R", "CCI.CCIIR"), "xts")
ts_ggplot(ts_scale(ts_span(
  ts_c(
    'GDP Growth' = ts_pc(dta[, 'GDP.PBRTT.A.R']),
    'Consumer Sentiment Index' = dta[, 'CCI.CCIIR']
  ),
  start = "1995-01-01"
))) +
ggplot2::ggtitle("GDP and Consumer Sentiment", subtitle = "normalized values") +
  theme_tsbox() +
  scale_color_tsbox()

## End(Not run)
```

---

**ts_index**  
*Indices from Levels or Percentage Rates*

**Description**

`ts_index` returns an index series, with value of 1 at base date. `ts_compound` builds an index from percentage change rates, starting with 1 and compounding the rates.
Usage

\texttt{ts\_compound(x, denominator = 100)}

\texttt{ts\_index(x, base = NULL)}

Arguments

\texttt{x} \hspace{1cm} ts-boxable time series, an object of class \texttt{ts}, \texttt{xts}, \texttt{zoo}, \texttt{data.frame}, \texttt{data.table}, \texttt{tbl}, \texttt{tbl\_ts}, \texttt{tbl\_time}, \texttt{tis}, \texttt{irts} or \texttt{timeSeries}.

\texttt{denominator} \hspace{1cm} numeric, set equal to one if percentage change rate is given a decimal fraction

\texttt{base} \hspace{1cm} base date, character string, \texttt{Date} or \texttt{POSIXct}, at which the

Value

a ts-boxable time series, with the same class as the input.

Examples

\begin{verbatim}
head(ts_compound(ts_pc(ts_c(fdeaths, mdeaths))))
head(ts_index(ts_df(ts_c(fdeaths, mdeaths)), "1974-02-01"))

ts_plot(
  'My Expert Knowledge' = ts_chain(
    mdeaths,
    ts_compound(ts_bind(ts_pc(mdeaths), 15, 23, 33)),
    'So Far' = mdeaths,
    title = "A Very Manual Forecast"
)
\end{verbatim}

---

ts\_lag \hspace{1cm} \textit{Lag or Lead of Time Series}

Description

Shift time stamps in ts-boxable time series, either by a number of periods or by a fixed amount of time.

Usage

\texttt{ts\_lag(x, by = 1)}

Arguments

\texttt{x} \hspace{1cm} ts-boxable time series, an object of class \texttt{ts}, \texttt{xts}, \texttt{zoo}, \texttt{data.frame}, \texttt{data.table}, \texttt{tbl}, \texttt{tbl\_ts}, \texttt{tbl\_time}, \texttt{tis}, \texttt{irts} or \texttt{timeSeries}.

\texttt{by} \hspace{1cm} integer or character, either the number of shifting periods (integer), or an absolute amount of time (character). See details.
Details

The lag order, by, is defined the opposite way as in R base. Thus, -1 is a lead and +1 a lag.

If by is integer, the time stamp is shifted by the number of periods. This requires the series to be regular.

If by is character, the time stamp is shifted by a specific amount of time. This can be one of one of "sec", "min", "hour", "day", "week", "month", "quarter" or "year", optionally pre-
ceded by a (positive or negative) integer and a space, or followed by plural "s". This is passed to base::seq.Date(). This does not require the series to be regular.

Value

a ts-boxable time series, with the same class as the input. If time stamp shifting causes the object to be irregular, a data frame is returned.

Examples

ts_plot(AirPassengers, ts_lag(AirPassengers), title = "Illustrating the need for glasses")

head(ts_lag(fdeaths, "1 month"))
head(ts_lag(fdeaths, "1 year"))
head(ts_lag(ts_df(fdeaths), "2 day"))
head(ts_lag(ts_df(fdeaths), "2 min"))
head(ts_lag(ts_df(fdeaths), "-1 day"))

Description

Functions to reshape multiple time series from 'wide' to 'long' and vice versa. Note that long format data frames are ts-boxable objects, where wide format data frames are not. ts_long automatically identifies a time column, and uses columns on the left as id columns.

Usage

   ts_long(x)
   ts_wide(x)

Arguments

   x       a ts-boxable time series, or a wide data.frame, data.table, or tibble.

Value

   object with the same class as input
Examples

```
df.wide <- ts_wide(ts_df(ts_c(mdeaths, fdeaths)))
head(df.wide)
head(ts_long(df.wide))
```

---

**ts_na_omit**

*Omit NA values*

Description

Remove NA values in ts-boxable objects, turning explicit into implicit missing values.

Usage

```
ws_na_omit(x)
```

Arguments

- **x**: ts-boxable time series, an object of class `ts`, `xts`, `zoo`, `data.frame`, `data.table`, `tbl`, `tbl_ts`, `tbl_time`, `tis`, `irts` or `timeSeries`.

Details

Note that internal NAs in `ts` time series will not be removed, as this conflicts with the regular structure.

Value

a ts-boxable time series, with the same class as the input.

See Also

- `ts_regular`, for the opposite, turning implicit into explicit missing values.

Examples

```
x <- AirPassengers
x[c(2, 4)] <- NA

# A ts object does only know explicit NAs
head(ts_na_omit(x))

# by default, NAs are implicit in data frames
head(ts_df(x))

# make NAs explicit
head(ts_regular(ts_df(x)))

# and implicit again
head(ts_na_omit(ts_regular(ts_df(x))))
```
Descriptive Statistics

First Differences and Percentage Change Rates

Description

`ts_pcy` and `ts_diffy` calculate the percentage change rate and the difference compared to the previous period, `ts_pcy` and `ts_diffy` calculate the percentage change rate compared to the same period of the previous year. `ts_pca` calculates annualized percentage change rates compared to the previous period.

Usage

`ts_pc(x)`
`ts_diff(x)`
`ts_pca(x)`
`ts_pcy(x)`
`ts_diffy(x)`

Arguments

`x` ts-boxable time series, an object of class `ts`, `xts`, `zoo`, `data.frame`, `data.table`, `tbl`, `tbl_ts`, `tbl_time`, `tis`, `irts` or `timeSeries`.

Value

A ts-boxable time series, with the same class as the input.

Examples

```
tail(ts_diff(ts_c(fdeaths, mdeaths)))
tail(ts_pc(ts_c(fdeaths, mdeaths)))
tail(ts_pca(ts_c(fdeaths, mdeaths)))
tail(ts_pcy(ts_c(fdeaths, mdeaths)))
tail(ts_diffy(ts_c(fdeaths, mdeaths)))
```
ts_pick

Pick Series (Experimental)

Description

Pick (and optionally rename) series from multiple time series.

Usage

ts_pick(x, ...)

Arguments

x ts-boxable time series, an object of class ts, xts, zoo, data.frame, data.table, tbl, tbl_ts, tbl_time, tis, irts or timeSeries.

... character string(s), names of the series to be picked, or integer, with positions. If arguments are named, the series will be renamed.

Value

a ts-boxable time series, with the same class as the input.

Examples

# Interactive use

```
# Interactive use

ts_plot(ts_pick(
    EuStockMarkets,
    `My Dax` = "DAX",
    `My Smi` = "SMI"
))
head(ts_pick(EuStockMarkets, c(1, 2)))
head(ts_pick(EuStockMarkets, `My Dax` = 'DAX', `My Smi` = 'SMI'))
```

# Programming use

```
# Programming use
to.be.picked.and.renamed <- c('My Dax' = "DAX", 'My Smi' = "SMI")
head(ts_pick(EuStockMarkets, to.be.picked.and.renamed))
```
ts_plot

Plot Time Series

Description

\texttt{ts_plot()} is a fast and simple plotting function for ts-boxable time series, with limited customizability. For more theme options, use \texttt{ts_ggplot()}.

Usage

\begin{verbatim}
\texttt{ts_plot(..., title, subtitle, ylab = \"\", family = getOption("ts_font", "sans"))}
\end{verbatim}

Arguments

\begin{itemize}
  \item \ldots\quad ts-boxable time series, objects of class \texttt{ts}, \texttt{xts}, \texttt{data.frame}, \texttt{data.table}, or \texttt{tibble}.
  \item \texttt{title}\quad title (optional)
  \item \texttt{subtitle}\quad subtitle (optional)
  \item \texttt{ylab}\quad ylab (optional)
  \item \texttt{family}\quad font family (optional, can also be set via options)
\end{itemize}

Details

Both \texttt{ts_plot()} and \texttt{ts_ggplot()} combine multiple ID dimensions into a single dimension. To plot multiple dimensions in different shapes, facets, etc., use standard \texttt{ggplot}.

Limited customizability of \texttt{ts_plot} is available via options. See examples.

See Also

\texttt{ts_ggplot()}, for a plotting function based on \texttt{ggplot2}. \texttt{ts_dygraphs()}, for interactive time series plots. \texttt{ts_save()} to save a plot to the file system.

Examples

\begin{verbatim}
\texttt{ts_plot(}
  \texttt{AirPassengers,}
  \texttt{title = \"Airline passengers\",}
  \texttt{subtitle = \"The classic Box & Jenkins airline data\")}
\end{verbatim}

# naming arguments
\texttt{ts_plot(total = ldeaths, female = fdeaths, male = mdeaths)}

# using different ts-boxable objects
\texttt{ts_plot(ts_scale(ts_c(}
ts_regular

Enforce Regularity

Description
Enforces regularity in data frame and xts objects, by turning implicit NAs into explicit NAs. In ts objects, regularity is automatically enforced.

Usage
```
.ts_regular(x, fill = NA)
```

Arguments

- **x**: ts-boxable time series, an object of class ts, xts, zoo, data.frame, data.table, tbl, tbl_ts, tbl_time, tis, irts or timeSeries.
- **fill**: instead of NA, an alternative value can be specified

Examples
```
x0 <- AirPassengers
x0[c(10, 15)] <- NA
x <- ts_na_omit(ts_dts(x0))
.ts_regular(x)
.ts_regular(x, fill = 0)

m <- mdeaths
m[c(10, 69)] <- NA
f <- fdeaths
f[c(1, 3, 15)] <- NA
.ts_regular(ts_na_omit(ts_dts(ts_c(f, m))))
```
ts_save

Save Previous Plot

Description
Save Previous Plot

Usage
```r
ts_save(filename = tempfile(fileext = ".pdf"), width = 10, height = 5, device = NULL, open = TRUE)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>filename</td>
</tr>
<tr>
<td>width</td>
<td>width</td>
</tr>
<tr>
<td>height</td>
<td>height</td>
</tr>
<tr>
<td>device</td>
<td>device</td>
</tr>
<tr>
<td>open</td>
<td>logical, should the saved plot be opened?</td>
</tr>
</tbody>
</table>

ts_scale

Normalized Time Series

Description
Subtract mean and divide by standard deviation. Based on `base::scale()`.

Usage
```r
ts_scale(x, center = TRUE, scale = TRUE)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>ts_boxable time series</td>
</tr>
<tr>
<td>center</td>
<td>logical</td>
</tr>
<tr>
<td>scale</td>
<td>logical</td>
</tr>
</tbody>
</table>

Examples
```r
ts_plot(ts_scale((ts_c(airmiles, co2, JohnsonJohnson, discoveries))))
ts_plot(ts_scale(ts_c(AirPassengers, DAX = EuStockMarkets[, 'DAX'])))
```
Limit Time Span

Description

Filter time series for a time span.

Usage

\[
\text{ts\_span}(x, \text{start} = \text{NULL}, \text{end} = \text{NULL}, \text{template} = \text{NULL}, \\
\text{extend} = \text{FALSE})
\]

Arguments

- **x**: ts-boxable time series, an object of class ts, xts, zoo, data.frame, data.table, tbl, tbl_ts, tbl_time, tis, irts or timeSeries.
- **start**: start date, character string, Date or POSIXct
- **end**: end date, character string, Date or POSIXct
- **template**: ts-boxable time series, an object of class ts, xts, data.frame, data.table, or tibble. If provided, from and to will be extracted from the object.
- **extend**: logical. If true, the start and end values are allowed to extend the series (by adding NA values).

Details

All date and times, when entered as character strings, are processed by anytime::anydate() or anytime::anytime(). Thus a wide range of inputs are possible. See examples.

start and end can be specified relative to each other, using one of "sec", "min", "hour", "day", "week", "month", "quarter" or "year", or an abbreviation. If the series are of the same frequency, the shift can be specified in periods. See examples.

Value

a ts-boxable time series, with the same class as the input.

Examples

# use 'anytime' shortcuts
\[
\text{ts\_span(mdeaths, start = "1979")} \quad \# \text{shortcut for 1979-01-01}
\]

\[
\text{ts\_span(mdeaths, start = "1979-4")} \quad \# \text{shortcut for 1979-04-01}
\]

\[
\text{ts\_span(mdeaths, start = "197904")} \quad \# \text{shortcut for 1979-04-01}
\]

# it's fine to use an to date outside of series span
\[
\text{ts\_span(mdeaths, end = "2001-01-01")}
\]
ts_summary

# use strings to set start or end relative to each other
ts_span(mdeaths, start = "-7 month")  # last 7 months
ts_span(mdeaths, start = -7)          # last 7 periods
ts_span(mdeaths, start = -1)         # last single value
ts_span(mdeaths, end = "1e4 hours")  # first 10000 hours

ts_plot(
  ts_span(mdeaths, start = "-3 years"),
  title = "Three years ago",
  subtitle = "The last three years of available data"
)

ts_ggplot(
  ts_span(mdeaths, end = "28 weeks"),
  title = "28 weeks later",
  subtitle = "The first 28 weeks of available data"
) + theme_tsbox() + scale_color_tsbox()

# Limit span of 'discoveries' to the same span as 'AirPassengers'
ts_span(discoveries, template = AirPassengers)
ts_span(mdeaths, end = "19801201", extend = TRUE)

---

### ts_summary

#### Time Series Properties

**Description**

Extract time series properties, such as the number of observations (obs), the time differences between observations (obs), the number of observations per year (freq), and the start time stamp (start) and the end time stamp (end) of the series.

**Usage**

```r
ts_summary(x, spark = FALSE)
```

**Arguments**

- `x`  
  ts-boxable time series, an object of class `ts`, `xts`, `zoo`, `data.frame`, `data.table`, `tbl`, `tbl_ts`, `tbl_time`, `tis`, `irts` or `timeSeries`.

- `spark`  
  logical should an additional column with a spark-line added to the data frame (experimental, ASCII only on Windows.)

**Value**

`ts_summary` returns a `data.frame`. Individual column can be accessed through the $ notation (see examples).
Examples

```
ts_summary(ts_c(mdeaths, austres))
ts_summary(ts_c(mdeaths, austres), spark = TRUE)
# Extracting specific properties
ts_summary(AirPassengers)$start
ts_summary(AirPassengers)$freq
ts_summary(AirPassengers)$obs
```

---

**ts_trend**  
*Loess Trend Estimation*

**Description**

Trend estimation that uses `stats::loess()`.

**Usage**

```
ts_trend(x, ...)
```

**Arguments**

- `x`  
  ts-boxable time series, an object of class `ts`, `xts`, `zoo`, `data.frame`, `data.table`, `tbl`, `tbl_ts`, `tbl_time`, `tis`, `irts` or `timeSeries`.

- `...`  
  arguments, passed to `stats::loess()`:
  - `degree` degree of Loess smoothing
  - `span` smoothing parameter, if NULL, an automated search performed (see Details)

**Examples**

```
  ts_plot(
    `Raw series` = fdeaths,  
    `Loess trend` = ts_trend(fdeaths),
    title = "Deaths from Lung Diseases",  
    subtitle = "per month"
  )
```
Description

tsbox is built around a set of converters, which convert time series stored as ts, xts, data.frame, data.table or tibble to each other.

Usage

```
  ts_data.frame(x)
  ts_df(x)
  ts_data.table(x)
  ts_dt(x)
  ts_tbl(x)
  ts_tibbletime(x)
  ts_timeSeries(x)
  ts_tis(x)
  ts_ts(x)
  ts_tibble(x)
  ts_tsibble(x)
  ts_tslist(x)
  ts_xts(x)
  ts_zoo(x)
```

Arguments

```
x           ts-boxable time series, an object of class ts, xts, zoo, data.frame, data.table, tbl, tbl_ts, tbl_time, tis, irts or timeSeries.
```

Details

In data frames, multiple time series will be stored in a 'long' format. tsbox detects a value, a time and zero to several id columns. Column detection is done in the following order:
1. Starting **on the right**, the first first numeric or integer column is used as **value column**.

2. Using the remaining columns, and starting on the right again, the first Date, POSIXct, numeric or character column is used as **time column**. Character strings are parsed by `anytime::anytime()`.
   The time stamp, time, indicates the beginning of a period.

3. All remaining columns are **id columns**. Each unique combination of id columns points to a time series.

**Alternatively**, the **time** column and the **value** column to be explicitly named as time and value. If explicit names are used, the column order will be ignored.

Whenever possible, tsbox relies on **heuristic time conversion**. When a monthly "ts" time series, e.g., AirPassengers, is converted to a data frame, each time stamp (of class "Date") is the first day of the month. In most circumstances, this reflects the actual meaning of the data stored in a "ts" object. Technically, of course, this is not correct: "ts" objects divide time in period of equal length, while in reality, February is shorter than January. Heuristic conversion is done for frequencies of 0.1 (decades), 1 (years), 4 (quarters) and 12 (month).

For other frequencies, e.g., 260, of EuStockMarkets, tsbox uses **exact time conversion**. The year is divided into 260 equally long units, and time stamp of a period will be a point in time (of class "POSIXct").

**Value**

- ts-boxable time series of the desired class, an object of class ts, xts, data.frame, data.table, or tibble.

**Examples**

```r
x.ts <- ts_c(mdeaths, fdeaths)
hed(x.ts)
hed(ts_df(x.ts))

suppressMessages(library(dplyr))
hed(ts_tbl(x.ts))

suppressMessages(library(data.table))
hed(ts_dt(x.ts))

suppressMessages(library(xts))
hed(ts_xts(x.ts))

# heuristic time conversion
# 1 month: approx. 1/12 year
head(ts_df(AirPassengers))

# exact time conversion
# 1 trading day: exactly 1/260 year
head(ts_df(EuStockMarkets))

# multiple id
multi.id.df <- rbind(
```

within(ts_df(ts_c(fdeaths, mdeaths)), type <- "level"),
within(ts_pc(ts_df(ts_c(fdeaths, mdeaths))), type <- "pc")
)
head(ts_ts(multi.id.df))
ts_plot(multi.id.df)
Index

*Topic package
  tsbox-package, 2
  %ts%(ts_arithmetic), 5
  %ts+(ts_arithmetic), 5
  %ts-(ts_arithmetic), 5
  %ts/(ts_arithmetic), 5

  anytime::anytime(), 26
  base::mean(), 11
  base::scale(), 21
  base::seq.Date(), 15

  colors_tsbox(ts_ggplot), 12
  copy_class, 3

  load_suggested(ts_), 4
  relevant_class, 3

  scale_color_tsbox(ts_ggplot), 12
  scale_color_tsbox(), 12
  scale_fill_tsbox(ts_ggplot), 12
  stats:::loess(), 24
  stats:::prcomp, 10

  theme_tsbox(ts_ggplot), 12
  theme_tsbox(), 12
  ts_, 4, 10
  ts_apply(ts_), 4
  ts_arithmetic, 5
  ts_bind, 6, 8
  ts_boxable, 7
  ts_c, 7, 8
  ts_chain(ts_bind), 6
  ts_compound(ts_index), 13
  ts_data.frame(ts_ts), 25
  ts_data.table(ts_ts), 25
  ts_default, 9
  ts_df(ts_ts), 25
  ts_diff(ts_ts), 25
  ts_diffy(ts_pc), 17
  ts_dt(ts_ts), 25
  ts_dts, 9
  ts_dygraphs(ts_examples), 10
  ts_dygraphs(), 13, 19
  ts_end(tsbox-deprecated), 4
  ts_examples, 5, 10
  ts_forecast(ts_examples), 10
  ts_frequency, 11
  ts_ggplot, 12
  ts_ggplot(), 19
  ts_index, 13
  ts_irts(ts_ts), 25
  ts_lag, 14
  ts_long, 15
  ts_na_omit, 16
  ts_pc, 17
  ts_pca(ts_pc), 17
  ts_pcy(ts_pc), 17
  ts_pick, 18
  ts_plot, 19
  ts_plot(), 12, 13
  ts_prcomp(ts_examples), 10
  ts_regular, 16, 20
  ts_save, 21
  ts_save(), 19
  ts_scale, 21
  ts_seas(ts_examples), 10
  ts_span, 22
  ts_start(tsbox-deprecated), 4
  ts_summary, 23
  ts_tbl(ts_ts), 25
  ts_tibbletime(ts_ts), 25
  ts_tis(ts_ts), 25
  ts_trend, 24
  ts_ts, 25
  ts_tsibble(ts_ts), 25
  ts_tslist(ts_ts), 25
INDEX

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ts_wide</td>
<td>(ts_long), 15</td>
</tr>
<tr>
<td>ts_xts</td>
<td>(ts_ts), 25</td>
</tr>
<tr>
<td>ts_zoo</td>
<td>(ts_ts), 25</td>
</tr>
<tr>
<td>tsbox</td>
<td>(tsbox-package), 2</td>
</tr>
<tr>
<td>tsbox-deprecated</td>
<td>4</td>
</tr>
<tr>
<td>tsbox-package</td>
<td>2</td>
</tr>
</tbody>
</table>