Package: tstools (via r-universe)

August 25, 2024

Type Package

Title A Time Series Toolbox for Official Statistics

Version 0.4.3

Description Plot official statistics' time series conveniently: automatic legends, highlight windows, stacked bar chars with positive and negative contributions, sum-as-line option, two y-axes with automatic horizontal grids that fit both axes and other popular chart types. 'tstools' comes with a plethora of defaults to let you plot without setting an abundance of parameters first, but gives you the flexibility to tweak the defaults. In addition to charts, 'tstools' provides a super fast, 'data.table' backed time series I/O that allows the user to export / import long format, wide format and transposed wide format data to various file types.

License GPL-2

URL https://github.com/KOF-ch/tstools

BugReports https://github.com/KOF-ch/tstools/issues

Depends R (>= 3.0.0), zoo (>= 1.7-12)

Imports data.table, graphics, jsonlite, stats, xts, yaml

Suggests knitr, openxlsx, reshape2, rmarkdown, testthat

VignetteBuilder knitr

Encoding UTF-8

LazyData true

NeedsCompilation no

RoxygenNote 7.2.3

Repository https://kof-ch.r-universe.dev

RemoteUrl https://github.com/kof-ch/tstools

RemoteRef HEAD

RemoteSha 122c27b8b2a788ab1ecdfef9f96cef209fc90cd7

Contents

`.read_swissdata_meta_unknown_format` *Read Meta Data File w/o File Extension*

Description

Read a meta file without extension -> unknown format Tries to determine format (yaml, json) and return the metadata path must point to the file without extension e.g. swissdata_wd/set_id/set_id

Usage

```
.read_swissdata_meta_unknown_format(path)
```

Arguments

`path`	character file path.

Value

Meta list if file could be located, empty list otherwise

`CHGDP` *CH GDP Growth Contributions*

Description

A list of time series containing sector contributions to Swiss GDP over time.

Usage

```
CHGDP
```

Format

List list of six time series of class ts, containing contributions to Swiss GDP growth

manufacturing Growth contribution of manufacturing.

energy Growth contribution of energy, water sector

construction Growth contribution construction sector.

hotels Growth contribution of hotels.

fin_insur Growth contribution of financial services and insurances.

other Growth contribution of other sectors.

Source

https://www.seco.admin.ch/seco/en/home/wirtschaftslage---wirtschaftspolitik/Wirtschaftslage/bip-quartalsschaetzungen-/daten.html

---

`color_blind` *Provide Colorblind Compliant Colors*

Description

8 Hex RGB color defintions suitable for charts for colorblind people.

Usage

```
color_blind()
```

compute_decimal_time — *Compute Decimal Time from a ts Period Vector*

Description

Standard ts object use a vector of length two to store a period. E.g. 2010,1 means first quarter of 2010, if the series was quarterly and first month if the series was monthly etc.

Usage

```
compute_decimal_time(v, f)
```

Arguments

`v`	integer vector denoting a point in time
`f`	frequency

concat_ts — *Concatenate to Non-Overlapping Time Series*

Description

Append one time series to another. This only works for non-overlapping time series of the same frequency. For overlapping time series please see `resolveOverlap`.

Usage

```
concat_ts(ts1, ts2)
```

Arguments

`ts1`	object of class ts1, typically the older of two time series.
`ts2`	object of class ts1, typically the younger of two time series.

`create_cross_sec_overview` *Create an Overview data.table of (last) observations*

Description

Create a data.table that shows the i-th obsersvation of several time series.

Usage

```
create_cross_sec_overview(list_of_rows, col_labels, tsl, selected_period)
```

Arguments

`list_of_rows`	list of time series names
`col_labels`	character list of column labels
`tsl`	list of time series object to select from
`selected_period`	numeric date as in defining ts objects.

Examples

```
tsl <- generate_random_ts(10, lengths = 20)
list_of_rows <- list(
  "group 1" = c("ts1", "ts2", "ts3", "ts4"),
  "group 2" = c("ts5", "ts6", "ts7", "ts10")
)
# These are no real +,=,- values just random data.
create_cross_sec_overview(
  list_of_rows,
  c("+", "=", "-", "random"),
  tsl, c(1988, 12)
)
```

`create_dummy_ts` *Flexible Function to Create Time Series Dummy Variables*

Description

Generate time series with a default value that is changed within a certain subperiod. The function allows for additional convenience when specifying single period dummies and dummies that go from a certain point in time to the end of the series.

Usage

```
create_dummy_ts(
  end_basic,
  dummy_start,
  dummy_end = NULL,
  sp = T,
  start_basic = c(1980, 1),
  basic_value = 0,
  dummy_value = 1,
  frequency = 4
)
```

Arguments

`end_basic`	numeric vector of form c(yyyy,p) defining the end of the time series.
`dummy_start`	numeric vector of form c(yyyy,p) defining the beginning of the period with different value.
`dummy_end`	numeric vector of form c(yyyy,p) defining the end of the period with different value. Defaults to NULL, using the end_date of the series.
`sp`	logical should NULL value for dummy_end lead to a single period dummy (TRUE) or to alternative values until the end.
`start_basic`	numeric vector of form c(yyyy,p) defining the start of the time series. Defaults to c(1980,1)
`basic_value`	default value of the time series, defaults to 0.
`dummy_value`	the alternative value, defaults to 1.
`frequency`	integer frequency of the regular time series, defaults to 4 (quarterly).

Author(s)

Matthias Bannert

`df_to_reg_ts` *Turn data.frame to Regular Monthly or Quarterly Time Series*

Description

Turn a data.frame with date columns to a regular time series object if possible. Design to work with quarterly and monthly data.

Usage

```
df_to_reg_ts(
  dframe,
  var_cols,
  year_col = "year",
  period_col = "month",
  freq = 12,
  return_ts = T,
  by = NULL
)
```

Arguments

dframe	data.frame input
var_cols	columns that contain variables as opposed to date index.
year_col	integer, logical or character vector indicating the year position within the data.frame.
period_col	integer, logical or character vector indicating the period position within the data.frame.
freq	integer indicating the frequency of new time series.
return_ts	logical should a (list of) time series be returned? Defaults to TRUE. FALSE returns data.frame.
by	character overwrite automatically detected (from freq) by parameter. e.g. '1 day'. Defaults to NULL.

Examples

```
start_m <- as.Date("2017-01-01")
df_missing <- data.frame(
  date = seq(start_m, by = "2 months", length = 6),
  value = 1:6,
  another_value = letters[1:6],
  yet_another_col = letters[6:1]
)
df_to_reg_ts(df_missing, c("value", "another_value"))
df_to_reg_ts(df_missing, c("value", "another_value"), return_ts = FALSE)
```

---

fill_year_with_nas *Fill Up a Time Series with NAs*

Description

When plotting a time series you might want set the range of the plot a little wider than just the start and end date of the original series. This function add fills up the current period (typically year) with NA.

Usage

```
fill_year_with_nas(x, add_periods = 1, fill_up_start = FALSE)
```

Arguments

x	object of class ts
add_periods	integer periods to add.
fill_up_start	logical should start year be filled up? Defaults to FALSE.

---

generate_random_ts *Generate a list of random time series*

Description

Useful for development or generating easily reproducible examples

Usage

```
generate_random_ts(
  n = 1,
  lengths = 36,
  starts = 1988,
  frequencies = 12,
  ranges_min = -1,
  ranges_max = 1,
  shifts = 0,
  ts_names = sprintf("ts%d", 1:n),
  seed = 30042018,
  random_NAs = FALSE,
  random_NA_proportions = 0.1,
  normally_distributed = FALSE,
  normal_means = 0,
  normal_sds = 1,
  frequency_shifts = FALSE,
  frequency_shift_after = 0.5
)
```

Arguments

n	The number of ts objects to generate
lengths	The lengths of the time series
starts	The start points of the time series in single number notation (e.g. 1990.5)
frequencies	The frequencies of the time series
ranges_min	The minimum values of the time series (if normally_distributed == FALSE)
ranges_max	The maximum values of the time series (if normally_distributed == FALSE)

`shifts` The shifts of time series values per series

`ts_names` The names of the ts objects in the resulting list

`seed` The random seed to be used

`random_NAs` Whether or not to introcude NA values at random positions in the ts

`random_NA_proportions` The fraction of values to be replaced with NAs if random_NAs is TRUE for the series

`normally_distributed` Use normal distribution instead of uniform

`normal_means` The means to use for normal distribution. Ignored unless normally_distributed is set to TRUE.

`normal_sds` The sds to use for normal distribution. Ignored unless normally_distributed is set to TRUE.

`frequency_shifts` Introduce frequency shifts (from 4 to 12) in the ts

`frequency_shift_after` After what fraction of the ts to shift frequencies

Details

Except for n and ts_names, all parameters accept either a single value or a vector of values. If a single value is supplied, that value is used for all time series being generated. If a vector is supplied, its values will be used for the corresponding series (e.g. starts[1] is used for the first series, starts[2] for the second and so on). Vectors are recycled if n is larger than their length.

If a ts_names vector is supplied, it must have length n and must not contain duplicates.

Value

A list of ts objects

Examples

```
generate_random_ts()

generate_random_ts(n = 3, ranges_min = c(-10, 0, 10), ranges_max = 20, starts = 2011)
```

---

getCiLegendColors *Helper to calculate ci colors for legends*

Description

Helper to calculate ci colors for legends

Usage

```
getCiLegendColors(color, n = 1, alpha = NULL)
```

Arguments

`color`	The color of the ci band
`n`	The number if ci bands
`alpha`	The alpha/transparency of the ci band

Details

Color may be specified as either a named color or a hex value Transparency may be specified as a hex value, number 0-255 or number 0-1

Value

A vector of non-transparent colors that result from oberlaying color over pure white 1:n times

---

`get_date_vector` *Compute the Period Vector representation of a Decimal Time value*

Description

The period value will be rounded down to the nearest integer. This function is not vectorized so only a single value can be converted at a time.

Usage

```
get_date_vector(dtime, frq)
```

Arguments

`dtime`	numeric decimal time value denoting a point in time
`frq`	integer frequency

`init_tsplot_theme` *Initiate Default Theme*

Description

The `tsplot` methods provide a theme argument which is used to pass on a plethora of useful defaults. These defaults are essentially stored in a list. Sometimes the user may want to tweak some of these defaults while keeping most of them. Hence the init_tsplot_theme function create a fresh list object containing default values for lot of different layout parameters etc. By replacing single elements of the list and passing the entire list to the plot function, single aspects can be tweaked while keeping most defaults. Init defaultTheme does not need any parameters.

This function provides sensible defaults for margins, font size, line width etc. scaled to the dimensions of the output file.

Usage

```
init_tsplot_theme(
  auto_bottom_margin = FALSE,
  band_fill_color = c(ETH_Petrol = colors$ETH_Petrol$`100`, ETH_Petrol_60 =
   colors$ETH_Petrol$`60`, ETH_Petrol_40 = colors$ETH_Petrol$`40`, ETH_Petrol_20 =
   colors$ETH_Petrol$`20`, ETH_Purple = colors$ETH_Purple$`100`, ETH_Purple_60 =
    colors$ETH_Purple$`60`, ETH_Purple_40 = colors$ETH_Purple$`40`),
  bar_border = "#000000",
  bar_border_lwd = 1,
  bar_fill_color = c(ETH_Petrol = colors$ETH_Petrol$`100`, ETH_Petrol_60 =
   colors$ETH_Petrol$`60`, ETH_Petrol_40 = colors$ETH_Petrol$`40`, ETH_Petrol_20 =
   colors$ETH_Petrol$`20`, ETH_Purple = colors$ETH_Purple$`100`, ETH_Purple_60 =
    colors$ETH_Purple$`60`, ETH_Purple_40 = colors$ETH_Purple$`40`),
  bar_gap = 15,
  bar_group_gap = 30,
  ci_alpha = "44",
  ci_colors = line_colors,
  ci_legend_label = "%ci_value%% ci for %series%",
  default_bottom_margin = 15,
  fill_up_start = FALSE,
  fill_year_with_nas = TRUE,
  highlight_color = colors$ETH_Grey$`20`,
  highlight_window = FALSE,
  highlight_window_end = NA,
  highlight_window_freq = 4,
  highlight_window_start = NA,
  highlight_y_values = NA,
  highlight_y_lwd = 2,
  highlight_y_color = "#000000",
  label_pos = "mid",
  legend_all_left = FALSE,
  legend_box_size = 2,
  legend_col = 1,
  legend_font_size = 1,
  legend_intersp_x = 1,
  legend_intersp_y = 1,
  legend_margin_bottom = 5,
  legend_margin_top = 12,
  legend_seg.len = 2,
  line_colors = c(ETH_Green_60 = colors$ETH_Green$`60`, ETH_Green_100 =
   colors$ETH_Green$`100`, ETH_Petrol_20 = colors$ETH_Petrol$`20`, ETH_Purple_60 =
   colors$ETH_Purple$`60`, ETH_Petrol_60 = colors$ETH_Petrol$`60`, ETH_Purple_100 =
    colors$ETH_Purple$`100`, ETH_Petrol_100 = colors$ETH_Petrol$`100`),
  line_to_middle = TRUE,
  lty = 1,
  lwd = c(2, 3, 1, 4, 2, 4),
  lwd_box = 1.5,
  lwd_quarterly_ticks = 1,
```

```
lwd_x_axis = 1.5,
lwd_y_axis = 1.5,
lwd_y_ticks = 1.5,
lwd_yearly_ticks = 1.5,
margins = c(NA, 7, 12, 7),
NA_continue_line = FALSE,
output_wide = FALSE,
point_symbol = 1:18,
pointsize = 12,
preferred_y_gap_sizes = c(25, 20, 15, 10, 5, 2.5, 1, 0.5),
quarterly_ticks = TRUE,
range_must_not_cross_zero = TRUE,
show_left_y_axis = TRUE,
show_points = FALSE,
show_right_y_axis = TRUE,
show_x_axis = TRUE,
show_y_grids = TRUE,
subtitle_adj = 0,
subtitle_adj_r = 0.9,
subtitle_cex = 1,
subtitle_margin = 2,
subtitle_outer = FALSE,
subtitle_transform = "toupper",
sum_as_line = FALSE,
sum_legend = "sum",
sum_line_color = c(ETH_Petrol_100 = colors$ETH_Petrol$`100`),
sum_line_lty = 1,
sum_line_lwd = 3,
tcl_quarterly_ticks = -0.4,
tcl_y_ticks = -0.75,
tcl_yearly_ticks = -0.75,
title_adj = 0,
title_cex.main = 1,
title_margin = 5,
title_outer = FALSE,
title_transform = NA,
total_bar_margin_pct = 0.2,
use_bar_gap_in_groups = FALSE,
use_box = FALSE,
x_tick_dt = 1,
xaxs = "i",
y_grid_color = colors$ETH_Grey$`40`,
y_grid_count = c(5, 6, 8, 10),
y_grid_count_strict = FALSE,
y_las = 2,
y_range_min_size = NULL,
y_tick_force_integers = FALSE,
y_tick_margin = 0.15,
```

```
  yaxs = "i",
  yearly_ticks = TRUE
)

init_tsplot_print_theme(
  output_wide = FALSE,
  margins = c(NA, 10/if (output_wide) 1 + 1/3 else 1, 10, 7/if (output_wide) 1 + 1/3 else
    1),
  lwd = scale_theme_param_for_print(c(2, 3, 1, 4, 2, 4), if (output_wide) c(10 + 2/3, 6)
    else c(8, 6)),
  sum_line_lwd = scale_theme_param_for_print(3, if (output_wide) c(10 + 2/3, 6) else c(8,
    6)),
  lwd_box = scale_theme_param_for_print(1.5, if (output_wide) c(10 + 2/3, 6) else c(8,
    6)),
  lwd_x_axis = scale_theme_param_for_print(1.5, if (output_wide) c(10 + 2/3, 6) else c(8,
    6)),
  lwd_yearly_ticks = scale_theme_param_for_print(1.5, if (output_wide) c(10 + 2/3, 6)
    else c(8, 6)),
  lwd_quarterly_ticks = scale_theme_param_for_print(1, if (output_wide) c(10 + 2/3, 6)
    else c(8, 6)),
  lwd_y_axis = scale_theme_param_for_print(1.5, if (output_wide) c(10 + 2/3, 6) else c(8,
    6)),
  lwd_y_ticks = scale_theme_param_for_print(1.5, if (output_wide) c(10 + 2/3, 6) else
    c(8, 6)),
  legend_intersp_y = scale_theme_param_for_print(1, if (output_wide) c(10 + 2/3, 6) else
    c(8, 6)),
  legend_box_size = scale_theme_param_for_print(2, if (output_wide) c(10 + 2/3, 6) else
    c(8, 6)),
  legend_margin_top = 8,
  legend_margin_bottom = 3,
  legend_seg.len = scale_theme_param_for_print(2, if (output_wide) c(10 + 2/3, 6) else
    c(8, 6)),
  pointsize = scale_theme_param_for_print(12, if (output_wide) c(10 + 2/3, 6) else c(8,
    6)),
  ...
)
```

Arguments

`auto_bottom_margin`
: logical Should the bottom margin be automatically calculated? This will be overridden if margins[1] is not NA. Default FALSE

`band_fill_color`
: character vector of hex colors for the bands if left_as_band == TRUE.

`bar_border`
: character hex colors for the border around bars in bar charts.

`bar_border_lwd`
: numeric The line width of the borders of bars in barplots. Default 1

`bar_fill_color`
: character vector of hex colors for the bars if left_as_bar == TRUE

`bar_gap`
: numeric The width of the gap between bars, in % of space alloted to the bar.

bar_group_gap
: numeric The width of the gap between groups of bars if group_bar_chart is TRUE.

ci_alpha
: Numeric 0-255, numeric 0-1 or hey 00-FF, transparency of the confidence interval bands

ci_colors
: Named colors or hex values Colors of the confidence interval bands

ci_legend_label
: character A formatting template for how the ci bands should be labelled. May contain the placeholders. '%ci_value%' will be replaced with the ci label. '%series%' (will be replaced with the series name) exactly once. Defaults to '%ci_value% ci for %series%'

default_bottom_margin
: numeric The bottom margin to use when margins[1] is NA but neither auto_legend nor auto_bottom_margin are true. Default 3

fill_up_start
: logical shoule the start of the year also be filled? Has no effect if fill_year_with_nas == FALSE. Default FALSE

fill_year_with_nas
: logical should year be filled up with missing in order to plot the entire year on the axis. Defaults to TRUE,

highlight_color
: character hex color code of highlight background, defaults to "#e9e9e9".

highlight_window
: logical should a particular time span be highlighted by different background color. Defaults to FALSE.

highlight_window_end
: integer vector highlight window start position, defaults to NA.,

highlight_window_freq
: integer frequency of the higlight window defintion, defaults to 4.

highlight_window_start
: integer vector highlight window start position, defaults to NA.

highlight_y_values
: numeric Vector of y values to highlight with a bold line

highlight_y_lwd
: integer Line width of the lines to highlight y values

highlight_y_color
: character Color of the lines to highlight y values

label_pos
: character, denotes where the x-axis label is at. defaults to "mid", alternative value: "start".

legend_all_left
: logical Should all legend entries be drawn on the left side of the plot? Default FALSE

legend_box_size
: numeric The size of the squares denoting bar colors in the legend. Default 2

legend_col
: integer number of columns for the legend, defaults to 3.

legend_font_size
: numeric passed on to the `cex` parameter of `legend`, defaults to 1

legend_intersp_x
: numeric same as base legend parameter, defaults to 1

legend_intersp_y
: numeric same as base legend parameter, defaults to 1

legend_margin_bottom
: numeric Distance between bottom of legend and bottom of graphic in % of device height, default 5

legend_margin_top
: numeric Distance between bottom of plot and top of legends % of device height, defaults to 12

legend_seg.len
: numeric Length of the line segments in the legend. Default 2

line_colors
: character vector of hex colors for 6 lines.

line_to_middle
: logical try to put a line into the middle of the plot. defaults to TRUE.

lty
: integer vector line type defaults to 1.

lwd
: integer vector line width, defaults to c(2,3,1,4,2,4).

lwd_box
: numeric Line width of the box around the plot. Default 1.5

lwd_quarterly_ticks
: numeric, width of yearly ticks, defaults to 1.

lwd_x_axis
: numeric The line width of the x axis. Default 1.5

lwd_y_axis
: numeric The line width of the y axis. Default 1.5

lwd_y_ticks
: numeric Line width of the y ticks. Default 1.5

lwd_yearly_ticks
: numeric, width of yearly ticks, defaults to 1.5.

margins
: integer vector defaults to c(NA, 4, 3, 3) + 0.1. Set margins[1] to NA to automatically determine the bottom margin such that the legend fits (if either auto_legend or auto_bottom_margin are TRUE)

NA_continue_line
: boolean If true, NA values in time series are ignored and a contonuous line is drawn. Multiple values to turn this behavior on/off for indivitual series are supported. Default FALSE

output_wide
: logical Should the output file be in a wide format (16:9) or (4:3)? Only if output_format is not "plot". Default FALSE

point_symbol
: integer or character The symbol to use for marking data points. Multiple values can be supplied to set the symbol for each individual series See `pch` in `?par`. Default 1:18

pointsize
: Numeric Point size of text, in 1/72 of an inch

preferred_y_gap_sizes
: numeric c(25, 20, 15, 10, 5, 2.5, 1, 0.5),

quarterly_ticks
: logical, should quarterly ticks be shown. Defaults to TRUE.

range_must_not_cross_zero
: logical automatic range finders are forced to do not find ranges below zero. Defaults to TRUE.

`show_left_y_axis`
: logical: should left y axis be shown, defaults to TRUE.

`show_points`
: boolean Whether to draw the symbol specified by point_symbol at the data points. Multiple values can be supplied to enable/disable showing points for each individual series Default FALSE

`show_right_y_axis`
: logical: should left y axis be shown, defaults to TRUE.

`show_x_axis`
: locigal: should x axis be shown, defaults to TRUE

`show_y_grids`
: logical should y_grids by shown at all, defaults to TRUE.

`subtitle_adj`
: numeric same as base `plot` parameter, defaults to 0.

`subtitle_adj_r`
: numeric same as base `plot` parameter, defaults to .9

`subtitle_cex`
: numeric same as base `plot` parameter, defaults to 1.

`subtitle_margin`
: numeric How far above the plot the title is placed in % of the device height. Defaults to 2.

`subtitle_outer`
: logical same as base `plot` parameter, defaults to TRUE

`subtitle_transform`
: function to transform the subtitle, defaults to "toupper",

`sum_as_line`
: logical should the sum of stacked time series be displayed as a line on top of stacked bar charts. Defaults to FALSE,

`sum_legend`
: character Label for the sum line, defaults to "sum". Set to NULL to not label the line at all.

`sum_line_color`
: character hex color of of sum_as_line, defaults "#91056a".

`sum_line_lty`
: integer line type of sum_as_line, defaults to 1.

`sum_line_lwd`
: integer line width of sum_as_line, defaults to 3.

`tcl_quarterly_ticks`
: numeric, length of quarterly ticks. See tcl_yearly_ticks, defaults to -0.4

`tcl_y_ticks`
: numeric Length of y ticks, see `tcl_yearly_ticks`. Default -0.75

`tcl_yearly_ticks`
: numeric, length of yearly ticks. Analogous to `cex` for `axis`. defaults to -0.75.

`title_adj`
: numeric, same as base `plot` parameter, defaults to 0.

`title_cex.main`
: numeric, same as base `plot` parameter defaults to 1

`title_margin`
: numeric How far above the plot the title is placed in % of the device height. Default 8

`title_outer`
: logical, currently undocumented. Defaults to TRUE.

`title_transform`
: function to transform the title, defaults to NA.

`total_bar_margin_pct`
: numeric defintion as in base plot, defaults to "i", defaults to .2,

`use_bar_gap_in_groups`
: logical Should there be gaps of size bar_gap between the bars in a group if group_bar_chart = TRUE? Default FALSE

use_box	logical use a box around the plot.
x_tick_dt	numeric The distance between ticks on the x axis in years. The first tick will always be at the start of the plotted time series. Defaults to 1.
xaxs	character axis defintion as in base plot, defaults to "i".
y_grid_color	character hex color of grids. Defaults to gray "#CCCCCC".
y_grid_count	integer vector preferred y grid counts c(5,6,8,10).
y_grid_count_strict	logical should we strictly stick to preferred y grid count? Defaults to FALSE.
y_las	integer, same as base `plot` parameter defaults to 2.
y_range_min_size	= NULL ,
y_tick_force_integers	logical Should y ticks be forced (rounded down) to whole numbers? Default FALSE
y_tick_margin	numeric, minimal percentage of horizontal grid that needs to be clean, i.e., without lines or bars. Defaults to 0.15 (15 percent).
yaxs	character axis defintion as in base plot, defaults to "i".
yearly_ticks	logical, should yearly ticks be shown. Defaults to TRUE.
...	All the other arguments to `init_tsplot_thene`

Details

Themes are essentially list that contain `par` parameters. Below all items are listed, some of them with comments. The per-line parameters (`line_colors`, `lwd`, `lty`, `show_points`, `point_symbol`) are recycled if more time series than elements on the corresponding theme vectors are supplied. e.g. if four time series are plotted but only two line_colors are supplied, the first and third series have the first color, while the second and fourth series have the second color. The list contains the following elements:

Author(s)

Matthias Bannert

Examples

```
## Not run:
# create a list
data(KOF)
tt <- init_tsplot_theme()
# adjust a single element
tt$highlight_window <- TRUE
# pass the list to tsplot
tsplot(KOF$kofbarometer, theme = tt)
# for more theme examples check the vignette
vignette("tstools")

## End(Not run)
```

KOF *KOF Barometer - Swiss Business Cycle Indicator*

Description

A list of time series containing two time series the KOF Barometer and the growth of Swiss GDP over time. KOF Barometer is a monthly business cycle indicator computed by the KOF Swiss Economic Institute. The GDP growth rate is used as a reference series to the Barometer.

Usage

```
KOF
```

Format

A list of two time series of class ts

kofbarometer KOF Barometer Indicator'

reference Reference series to KOF Barometer, change in Swiss GDP compared to previous month

baro_point_fc Auto Arima point forecast of the KOF Barometer

baro_lo_80 Auto Arima 80 percent CI lower bound of the KOF Barometer forecast

baro_hi_80 Auto Arima 80 percent CI upper bound of the KOF Barometer forecast

baro_lo_95 Auto Arima 95 percent CI lower bound of the KOF Barometer forecast

baro_hi_95 Auto Arima 95 percent CI upper bound of the KOF Barometer forecast ...

Source

https://kof.ethz.ch/en/forecasts-and-indicators/indicators/kof-economic-barometer.html

long_to_ts *Transform a long format data.frame of time series to a tslist*

Description

The data.frame must have three columns "date", "value" and "series" (identifying the time series)

Usage

```
long_to_ts(
  data,
  keep_last_freq_only = FALSE,
  force_xts = FALSE,
  strip_nas = TRUE
)
```

Arguments

data	data.frame The data.frame to be transformed
keep_last_freq_only	in case there is a frequency change in a time series, should only the part of the series be returned that has the same frequency as the last observation. This is useful when data start out crappy and then stabilize
force_xts	logical
strip_nas	logical should NAs be stripped (no leading and trailing nas) ?

---

m_to_q — *Turn monthly series with regular NAs to quarter*

Description

Monthly series with NAs in non-quarter months are turned to quarterly series. Series without NAs are just returned.

Usage

```
m_to_q(series)
```

Arguments

series	an object of class ts with monthly frequency

---

overlap_sorted_ts_lists — *Concat Time Series list wise*

Description

Concat overlapping time series list wise. List needs to be of same length. Takes names of list B.

Usage

```
overlap_sorted_ts_lists(listA, listB)
```

Arguments

listA	list of time series
listB	list of time series

overlap_ts_lists_by_name *Resolve Overlap Listwise, helpful with SA*

Description

Resolve Overlap Listwise, helpful with SA

Usage

```
overlap_ts_lists_by_name(listA, listB, chunkA = "_f4", chunkB = "_f12")
```

Arguments

listA	list of time series often of lower frequency
listB	list of time series often of higher frequency
chunkA	character chunk representing frequencies, defaults to _f4.
chunkB	character chunk representing frequences, defaults to _f12.

read_swissdata *Read data generated by the Swissdata project*

Description

Read data from swissdata compliant .csv files and turn them into a list of time series.

Usage

```
read_swissdata(
  path,
  key_columns = NULL,
  filter = NULL,
  aggregates = NULL,
  keep_last_freq_only = FALSE
)
```

Arguments

path	character full path to dataset.
key_columns	character vector specifying all columns that should be part of the key. Defaults to the dim.order specified by swissdata.
filter	function A function that is applied to the raw data.data table after it is read. Useful for filtering out undesired data.

`aggregates`	list A list of dimensions over which to aggregate data. The names of this list determing which function is used to calculate the aggregate (e.g. sum, mean etc.). Defaults to sum.
`keep_last_freq_only`	in case there is a frequency change in a time series, should only the part of the series be returned that has the same frequency as the last observation. This is useful when data start out crappy and then stabilize

Details

The order of dimensions in key_columns determines their order in the key The resulting ts_key will be of the form <swissdata-set-name>.<instance of key_columns[1]>...

Examples

```
ds_location <- system.file("example_data/ch.seco.css.csv", package = "tstools")
tslist <- read_swissdata(ds_location, "idx_type")
tsplot(tslist[1])
```

`read_swissdata_meta` *Read swissdata style yaml timeseries metadata*

Description

read_swissdata_meta reads the given .yaml file and converts it into a per-timeseries format.

Usage

```
read_swissdata_meta(path, locale = "de", as_list = FALSE)
```

Arguments

`path`	Path to the yaml file to be read
`locale`	Locale in which to read the data (supported are "de", "fr", "it" and "en")
`as_list`	Should the output be converted to a list?

Details

If as_list is set to TRUE, the function returns a nested list with one element per timeseries, otherwise a data.table with one row per series.

read_ts *Import time series data from a file.*

Description

If importing from a zip file, the archive should contain a single file with the extension .csv, .xlsx or .json.

Usage

```
read_ts(
  file,
  format = c("csv", "xlsx", "json", "zip"),
  sep = ",",
  skip = 0,
  column_names = c("date", "value", "series"),
  keep_last_freq_only = FALSE,
  force_xts = FALSE
)
```

Arguments

- `file` Path to the file to be read
- `format` Which file format is the data stored in? If no format is supplied, read_ts will attempt to guess from the file extension.
- `sep` character seperator for csv files. defaults to ','.
- `skip` numeric See data.table's fread.
- `column_names` character vector denoting column names, defaults to c("date","value","series).
- `keep_last_freq_only` in case there is a frequency change in a time series, should only the part of the series be returned that has the same frequency as the last observation. This is useful when data start out crappy and then stabilize after a while. Defaults to FALSE. Hence only the last part of the series is returned.
- `force_xts` If set to true, the time series will be returned as xts objects regargless of regularity. Setting this to TRUE means keep_last_freq_only is ignored.

Value

A named list of ts objects

`regularize` *Turn an Irregular Time Series to a Regular, ts-Based Series*

Description

Adds missing values to turn an irregular time series into a regular one. This function is currently experimental. Only works or target frequencies 1,2,4,12.

Usage

```
regularize(x)
```

Arguments

x	an irregular time series object of class zoo or xts.

Examples

```
ts1 <- rnorm(5)
dv <- c(
  seq(as.Date("2010-01-01"), length = 3, by = "3 years"),
  seq(as.Date("2018-01-01"), length = 2, by = "2 years")
)
library(zoo)
xx <- zoo(ts1, dv)
regularize(xx)

dv2 <- c(seq(as.Date("2010-01-01"), length = 20, by = "1 months"))
dv2 <- dv2[c(1:10, 14:20)]
xx2 <- zoo(rnorm(length(dv2)), dv2)
regularize(xx2)
```

`resolve_ts_overlap` *Concatenate Time Series and Resolve Overlap Automatically*

Description

Append time series to each other. Resolve overlap determines which of two ts class time series is reaching further and arranges the two series into first and second series accordingly. Both time series are concatenated to one if both series had the same frequency. Typically this function is used concatenate two series that have a certain overlap, but one series clearly starts earlier while the other lasts longer. If one series starts earlier and stops later, all elements of the shorter series will be inserted into the larger series, i.e. elements of the smaller series will replace the elements of the longer series. Usually ts2 is kept.

Usage

```
resolve_ts_overlap(ts1, ts2, keep_ts2 = T, tolerance = 0.001)
```

Arguments

`ts1`	ts time series, typically the older series
`ts2`	ts time series, typically the younger series
`keep_ts2`	logical should ts2 be kept? Defaults to TRUE.
`tolerance`	numeric when comparing min and max values with a index vector of a time series R runs in to trouble with precision handling, thus a tolerance needs to be set. Typically this does not need to be adjusted. E.g. 2010 != 2010.000. With the help of the tolerance parameter these two are equal.

Examples

```
ts1 <- ts(rnorm(100), start = c(1990, 1), frequency = 4)
ts2 <- ts(1:18, start = c(2000, 1), frequency = 4)
resolve_ts_overlap(ts1, ts2)

# automatical detection of correction sequence!
ts1 <- ts(rnorm(90), start = c(1990, 1), frequency = 4)
ts2 <- ts(1:60, start = c(2000, 1), frequency = 4)
resolve_ts_overlap(ts1, ts2)

# both series are of the same length use sequence of arguments.
ts1 <- ts(rnorm(100), start = c(1990, 1), frequency = 4)
ts2 <- ts(1:48, start = c(2003, 1), frequency = 4)
resolve_ts_overlap(ts1, ts2)
ts1 <- ts(rnorm(101), start = c(1990, 1), frequency = 4)
ts2 <- ts(1:61, start = c(2000, 1), frequency = 4)
resolve_ts_overlap(ts1, ts2)
#' clearly dominatn ts2 series
ts1 <- ts(rnorm(50), start = c(1990, 1), frequency = 4)
ts2 <- ts(1:100, start = c(1990, 1), frequency = 4)
resolve_ts_overlap(ts1, ts2)
```

`set_month_to_NA` *Set Periods to NA*

Description

This function is typically used to discard information in non-quarter month. I.e., data is only kept in January, April, July and December and otherwise set to NA. In combination with `m_to_q` this function is useful to turn monthly series into quarterly series by letting the quarter month values represent the entire quarter. This can be useful when data was interpolated because of mixing data of different frequencies and needs to be converted back to a regular, quarterly time series.

Usage

```
set_month_to_NA(series, keep_month = c(1, 4, 7, 10))
```

Arguments

`series`	ts object
`keep_month`	integer vector denoting the months that not be set to NA. Defaults to c(1,4,7,10)

Examples

```
tsq <- ts(1:20, start = c(1990, 1), frequency = 4)
aa <- tsqm(tsq)
m_to_q(set_month_to_NA(aa))
```

`start_ts_after_internal_nas` *Start a Time Series after the Last Internal NA*

Description

Internal NAs can cause trouble for time series operations such as X-13-ARIMA SEATS seasonal adjustment. Often, internal NAs only occur at at the beginning of a time series. Thus an easy solution to the problem is to discard the initial part of the data which contains the NA values. This way only a small part of the information is lost as opposed to not being able to seasonally adjust an entire series.

Usage

```
start_ts_after_internal_nas(series)
```

Arguments

`series`	on object of class ts

See Also

`stripLeadingNAsFromTs`, `stripTrailingNAsFromTs`

Examples

```
ts1 <- 1:30
ts1[c(3, 6)] <- NA
ts1 <- ts(ts1, start = c(2000, 1), frequency = 4)
start_ts_after_internal_nas(ts1)
```

strip_ts_of_leading_nas *Strip Leading / Trailing NAs from a Time Series Object*

Description

Removes NAs to begin with and starts time series index at the first non-NA value.

Usage

```
strip_ts_of_leading_nas(s)

strip_ts_of_trailing_nas(s)
```

Arguments

s	an object of class ts.

tsplot *Plot Time Series*

Description

Conveniently plot time series.

Usage

```
tsplot(
  ...,
  tsr = NULL,
  ci = NULL,
  left_as_bar = FALSE,
  group_bar_chart = FALSE,
  relative_bar_chart = FALSE,
  left_as_band = FALSE,
  plot_title = NULL,
  plot_subtitle = NULL,
  plot_subtitle_r = NULL,
  find_ticks_function = "findTicks",
  overall_xlim = NULL,
  overall_ylim = NULL,
  manual_date_ticks = NULL,
  manual_value_ticks_l = NULL,
  manual_value_ticks_r = NULL,
  manual_ticks_x = NULL,
```

```
  theme = NULL,
  quiet = TRUE,
  auto_legend = TRUE,
  output_format = "plot",
  filename = "tsplot",
  close_graphics_device = TRUE
)
```

Arguments

`...` multiple objects of class ts or a list of time series. All objects passed through the ... parameter relate to the standard left y-axis.

`tsr` list of time series objects of class ts.

`ci` list of confidence intervals for time series

`left_as_bar` logical should the series that relate to the left bar be drawn as (stacked) bar charts?

`group_bar_chart` logical should a bar chart be grouped instead of stacked?

`relative_bar_chart` logical Should time series be normalized such that bars range from 0 to 1? Defaults to FALSE. That way every sub bar (time series) is related to the global max. Hence do not expect every single bar to reach 1. This works for stacked and grouped charts and does not change anything but the scale of the chart.

`left_as_band` logical Should the time series assigned to the left axis be displayed as stacked area charts?

`plot_title` character title to be added to the plot

`plot_subtitle` character subtitle to be added to the plot

`plot_subtitle_r` character second subtitle to be added at the top right

`find_ticks_function` function to compute ticks.

`overall_xlim` integer overall x-axis limits, defaults to NULL.

`overall_ylim` integer overall y-axis limits, defaults to NULL.

`manual_date_ticks` character vector of manual date ticks.

`manual_value_ticks_l` numeric vector, forcing ticks to the left y-axis

`manual_value_ticks_r` numeric vector, forcing ticks to the right y-axis

`manual_ticks_x` numeric vector, forcing ticks on the x axis

`theme` list of default plot output parameters. Defaults to NULL, which leads to `init_tsplot_theme` being called. Please see the vignette for details about tweaking themes.

`quiet` logical suppress output, defaults to TRUE.

`auto_legend` logical should legends be printed automatically, defaults to TRUE.

`output_format` character Should the plot be drawn on screen or written to a file? Possible values are "plot" for screen output and "pdf". Default "plot"

`filename` character Path to the file to be written if `output_format` is "pdf". Default "tsplot.pdf"

`close_graphics_device`
logical Should the graphics device of the output file be closed after `tsplot`? Set this to FALSE to be able to make modifications to the plot after `tsplot` finishes. Default TRUE

Details

The ci parameter is a 3-level list of the form list(ts1 = list(ci_value_1 = list(ub = upper_bound_ts_object, lb = lower_bound_ts_object), ...), ...)

See `vignette("tstools")` for details.

`tsqm` *Interpolate quarterly time series into monthly*

Description

Repeat quarterly variables two times to generate a monthly variable.

Usage

```
tsqm(qts)
```

Arguments

`qts` quarterly time series

Examples

```
tsq <- ts(1:20, start = c(1990, 1), frequency = 4)
tsqm(tsq)
```

tstools-deprecated — *Deprecated function(s) in tstools*

Description

These functions are provided for compatibility with older version of the tstools package. They may eventually be completely removed.

Arguments

`...`	Parameters to be passed to the modern version of the function

Details

`computeDecimalTime`	now a synonym for `compute_decimal_time`
`concatTs`	now a synonym for `concat_ts`
`fillupYearWitnNAs`	now a synonym for `fill_year_with_nas`
`importTimeSeries`	now a synonym for `read_ts`
`init_tsplot_theme`	now a synonym for `init_tsplot_theme`
`overlapSortedLists`	now a synonym for `overlap_sorted_ts_lists`
`overlapTslByName`	now a synonym for `overlap_ts_lists_by_name`
`resolveOverlap`	now a synonym for `resolve_ts_overlap`
`stripLeadingNAsFromTs`	now a synonym for `strip_ts_of_leading_nas`
`stripTrailingNAsFromTs`	now a synonym for `strip_ts_of_trailing_nas`
`writeTimeSeries`	now a synonym for `write_ts`

wide_to_ts — *Transform a wide format data.frame into a tslist*

Description

The time series in the data.frame may be stored either rowwise or columnswise. The identifying column must be called date (for columnwise) or series (for rowwise)

Usage

```
wide_to_ts(data, keep_last_freq_only = FALSE, force_xts = FALSE)
```

Arguments

`data`	data.frame The data.frame to be transformed
`keep_last_freq_only`	in case there is a frequency change in a time series, should only the part of the series be returned that has the same frequency as the last observation. This is useful when data start out crappy and then stabilize after a while. Defaults to FALSE. Hence only the last part of the series is returned.
`force_xts`	boolean force xts format? Defaults to FALSE.

`write_ts` *Export a list of time series to a file.*

Description

Export a list of time series to a file.

Usage

```
write_ts(
  tl,
  fname = NULL,
  format = "csv",
  date_format = NULL,
  timestamp_to_fn = FALSE,
  round_digits = NULL,
  rdata_varname = "tslist",
  ...
)
```

Arguments

`tl`	list of time series
`fname`	character file name. Defaults to NULL, displaying output on console. Set a file name without file extension in order to store a file. Default file names / location are not CRAN compliant which is why the file name defaults to NULL.
`format`	character denotes export formats. Defaults to .csv. "csv", "xlsx", "json", "rdata" are available. Spreadsheet formats like csv allow for further optional parameters.
`date_format`	character denotes the date format. Defaults to NULL. If set to null the default is used: Jan 2010.
`timestamp_to_fn`	If TRUE, the current date will be appended to the file name. Defaults to FALSE.
`round_digits`	integer, precision in digits.
`rdata_varname`	character name of the list of time series within the store RData. Defaults to "tslist".
`...`	additional arguments used by spedific formats.

Details

Additional arguments covered by `...`

Name	Effect	Format(s)
`wide`	Export data in a wide format (one column per series)	CSV, XLSX
`transpose`	Transpose exported data (one row per series)	CSV, XLSX, only if wide = TRUE
`zip`	If set to TRUE, the file is compressed into a zip archive after export	any

Index