

Package: hubVis (via r-universe)

November 20, 2024

Title Plotting methods for hub models output
Version 0.1.1
Description Plotting methods for hub models output.
License MIT + file LICENSE
Encoding UTF-8
Imports cli, dplyr, ggplot2, grDevices, hubUtils (>= 0.0.1), methods, plotly, purrr, RColorBrewer, scales, stats
Remotes hubverse-org/hubUtils, hubverse-org/hubData, hubverse-org/hubExamples
Suggests hubData, hubExamples (>= 0.0.0.9001), rmarkdown, testthat (>= 3.0.0), vdiff
Roxygen list(markdown = TRUE)
RoxygenNote 7.3.2
URL <https://github.com/hubverse-org/hubVis>,
<https://hubverse-org.github.io/hubVis/>
BugReports <https://github.com/hubverse-org/hubVis/issues>
Config/testthat/edition 3
Config/Needs/website hubverse-org/hubStyle
Config/pak/sysreqs git make libicu-dev libssl-dev
Repository <https://hubverse-org.r-universe.dev>
RemoteUrl <https://github.com/hubverse-org/hubVis>
RemoteRef v0.1.1
RemoteSha 9f15bbf56e733f8599ed56ac64cfa7c44fd98205

Contents

plot_step_ahead_model_output	2
Index	5

`plot_step_ahead_model_output`*Basic Plot for model outputs*

Description

Create a simple Plotly time-series plot for model projection outputs.

Usage

```
plot_step_ahead_model_output(  
  model_out_tbl,  
  target_data,  
  use_median_as_point = FALSE,  
  show_plot = TRUE,  
  plot_target = TRUE,  
  x_col_name = "target_date",  
  x_target_col_name = "date",  
  show_legend = TRUE,  
  facet = NULL,  
  facet_scales = "fixed",  
  facet_nrow = NULL,  
  facet_ncol = NULL,  
  facet_title = "top left",  
  interactive = TRUE,  
  fill_by = "model_id",  
  pal_color = "Set2",  
  one_color = "blue",  
  fill_transparency = 0.25,  
  intervals = c(0.5, 0.8, 0.95),  
  top_layer = "model_output",  
  title = NULL,  
  ens_color = NULL,  
  ens_name = NULL,  
  group = NULL  
)
```

Arguments

<code>model_out_tbl</code>	a <code>model_out_tbl</code> object, containing all the required columns including a column containing date information (<code>x_col_name</code> parameter) and a column value.
<code>target_data</code>	a <code>data.frame</code> object containing the target data, with a column containing date information (<code>x_target_col_name</code> parameter) and a column observation. Ignored, if <code>plot_target = FALSE</code> .

use_median_as_point	a Boolean for using median quantile as point in plot. Default to FALSE. If TRUE, will select first any median output type value and if no median value included in model_out_tbl; will select quantile = 0.5 output type value.
show_plot	a boolean for showing the plot. Default to TRUE.
plot_target	a boolean for showing the target data in the plot. Default to TRUE. Data used in the plot comes from the parameter target_data
x_col_name	column name containing the date information for all_plot and all_ens data frames, value will be map to the x-axis of the plot. By default, "target_date".
x_target_col_name	column name containing the date information for target_data data frame, value will be map to the x-axis of the plot. By default, "date".
show_legend	a boolean for showing the legend in the plot. Default to TRUE.
facet	a unique value corresponding as a task_id variable name (interpretable as facet option for ggplot)
facet_scales	argument for scales as in ggplot2::facet_wrap or equivalent to shareX, shareY in plotly::subplot . Default to "fixed" (x and y axes are shared).
facet_nrow	a numeric, number of rows in the layout.
facet_ncol	a numeric, number of columns in the layout (ignored in plotly::subplot)
facet_title	a string, position of each subplot tile (value associated with the facet parameter). "top right", "top left" (default), "bottom right", "bottom left" are the possible values, NULL to remove the title. For interactive plot only.
interactive	a boolean to output an "interactive" version of the plot (using Plotly) or a "static" plot (using ggplot2). By default, TRUE (interactive plot)
fill_by	name of a column for specifying colors and legend in plot. The pal_color parameter can be use to change the palette. Default to model_id.
pal_color	a character string for specifying the palette color in the plot. Please refer to RColorBrewer::display.brewer.all() . If NULL, only one_color parameter will be used for all models. Default to "Set2"
one_color	a character string for specifying the color in the plot if pal_color is set to NULL. Please refer to colors() for accepted color names. Default to "blue"
fill_transparency	numeric value used to set transparency of intervals. 0 means fully transparent, 1 means opaque. Default to 0.25
intervals	a vector of numeric values indicating which central prediction interval levels to plot. NULL means no interval levels. If not provided, it will default to c(.5, .8, .95). When plotting 6 models or more, the plot will be reduced to show .95 interval only. Value possibles: 0.5, 0.8, 0.9, 0.95
top_layer	character vector, where the first element indicates the top layer of the resulting plot. Possible options are "model_output" (default) and "target"
title	a character string, if not NULL, will be added as title to the plot
ens_color	a character string of a color name, if not NULL, will be use as color for the model name associated with the parameter ens_name (both parameter need to be provided)

<code>ens_name</code>	a character string of a model name, if not NULL, will be use to change the color for the model name, associated with the parameter <code>ens_color</code> (both parameter need to be provided)
<code>group</code>	column name for partitioning the data in the data according the the value in the column. Please refer to ggplot2::aes_group_order for more information. By default, NULL (no partitioning).

Examples

```
# Load and Prepare Data
# The package hubExmample contains example files, please consult the
# documentation associated with the package, for more information.
library(hubExamples)
head(scenario_outputs)
head(scenario_target_ts)
projection_data <- dplyr::mutate(scenario_outputs,
  target_date = as.Date(origin_date) + (horizon * 7) - 1)
projection_data <- dplyr::filter(projection_data,
  scenario_id == "A-2021-03-05", location == "US")
projection_data <- hubUtils::as_model_out_tbl(projection_data)

target_data_us <- dplyr::filter(scenario_target_ts, location == "US",
  date < min(projection_data$target_date) + 21,
  date > "2020-10-01")

# Plot
plot_step_ahead_model_output(projection_data, target_data_us)
```

Index

`colors()`, [3](#)

`ggplot2::aes_group_order`, [4](#)

`ggplot2::facet_wrap`, [3](#)

`plot_step_ahead_model_output`, [2](#)

`plotly::subplot`, [3](#)

`RColorBrewer::display.brewer.all()`, [3](#)