Package 'TSplotly'

August 2, 2019

Type Package

Title Create Interactive Plots on Time Series Dataset

Version 1.1.1

URL http://socr.umich.edu/people/

Maintainer Yongkai Qiu <yongkai@umich.edu>

Description To better visualize time-series dataset,
'TSplotly' package provides an effective mechanism to utilize
the powerful 'plotly' package for graphing
time series data. It contains 5 core functions:
TSplot(): create interactive plot on time series data or fitted
ARIMA(X) models.
ADDline(): add lines on existing 'TSplot()' objects, as needed.
GGtoPY(): create a convenient way to transform (reformat) 'ggplot2'
datasets into a format that can work on 'plot_ly()'.
GTSplot(): create multiple 'plot_ly()' lines (time-series) based on
data frames containing multiple time-series data.
TSplot_gen(): a more general version of function 'TSplot()' that
can work on any time format.

Depends R (>= 3.4.0)

Imports forecast, plotly, zoo, ggplot2, dcemriS4, prettydoc

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

Author Yongkai Qiu [aut, cre],
Zhe Yin [aut],
Ivo Dinov [aut],
SOCR team [aut]

Suggests knitr, rmarkdown

VignetteBuilder knitr

Repository CRAN

Date/Publication 2019-08-02 11:00:08 UTC

R topics documented:

ADDline 2
GtoP_trans 3
GTSplot 5
MCSI_Data_monthAvg_melt 7
MCSI_Data_monthAvg_ts_Y 7
MCSI_Data_monthAvg_ts_Y_test 8
modArima_train 8
TSplot 9
TSplot_gen 10
X 12
xA_fMRI_1D_x20_y20_z11 12
xB_fMRI_1D_x30_y30_z13 13
xC_fMRI_1D_x40_y40_z12 13
X_new 14
X_test 14
Y 14

Index **15**

`ADDline` *'ADDline'*

Description

create a list of line data based on ARIMA(X) predicted result or ts function result(i.e. time series data)

Usage

```
ADDline(ARIMAmodel = NULL, XREG = NULL, TS = NULL, linetype = "TS",
  Name = NULL)
```

Arguments

`ARIMAmodel`	ARIMA model created by function `auto.arima()`
`XREG`	if using ARIMAX model, put in the regularized X matrix
`TS`	data created by `ts()` function
`linetype`	"TS" for time series data, "ARIMA" for ARIMA(X) predicted data
`Name`	title for this line

Details

This function ADDline is used to conduct a data transformation. It can take in original time-series data generated by function ts or fitted ARIMA(X) model generated by function auto.arima then take out four elements from those result to create a list which contains content that can be put in function add_lines or add_trace. So that we could add new lines in our plotly plot more easily.

Value

a list contains 4 elements:

X	data put in parameter "x" of plot_ly function
TEXT	rename the lable of variable X
Y	data put in parameter "y" of plot_ly function
NAME	title of the new line

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
require(forecast)
require(zoo)
require(plotly)

#Firstly create a base plotly plot
Tempplot<-TSplot(48,modArima_train,as.matrix(X_test),title_size = 8,
ts_original = "Original time series",ts_forecast = "Predicted time series")

# Generate a new line with ADDline function
newline<-ADDline(TS = MCSI_Data_monthAvg_ts_Y_test,linetype = "TS",Name = "Original Result")

## Put the new line into our plot
Tempplot%>%
 add_lines(x=newline$X,text=newline$TEXT,y=newline$Y,name=newline$NAME,line=list(color="grey"))
```

GtoP_trans *'GtoP_trans'*

Description

a dataset transformation method, change ggplot2 dataset into dataset that can be used by plot_ly

Usage

```
GtoP_trans(dataframe, NAME = NULL, X = NULL, Y = NULL)
```

Arguments

dataframe	a data frame data that can be used on ggplot2
NAME	column that will be used on creating different lines on plot_ly
X	column that will serve as the x axis on plot_ly
Y	column that will serve as the value(y) on plot_ly

Details

The function GtoP_trans is used to conduct a data transformation. Usually dataset working on ggplot2 is a kind of dataset that has long rows and short columns. And we may wish to create plot based on different levels for a particular column. However, this dataset won't work for plot_ly as it requires each level as a column. So this method will separate different levels in a column to create a new dataset with multiple columns representing the original levels.

Value

a data frame with each column serve as a line in plot_ly

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
require(ggplot2)
require(plotly)
require(zoo)
# Dataset in this example. We wish to generate plot with different lines based on variable "series".
# However, this dataset now can only be used on ggplot2
head(MCSI_Data_monthAvg_melt)

# A ggplot2 example here:
ggplot(MCSI_Data_monthAvg_melt[MCSI_Data_monthAvg_melt$series!="INCOME", ],
       aes(YYYYMM, value)) +
  geom_line(aes(linetype=series, colour = series), size=2) +
  geom_point(aes(shape=series, colour = series), size=0.3) +
  geom_smooth(aes(colour = series), se = TRUE) +
  coord_trans(y="log10") +
  xlab("Time (monthly)") + ylab("Index Values (log-scale)") +
  scale_x_date(date_breaks = "12 month", date_labels =  "%m-%Y")  +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        text = element_text(size=20))+ theme(legend.position="top")

# Change the dataset with our function
PYdf<-GtoP_trans(MCSI_Data_monthAvg_melt[MCSI_Data_monthAvg_melt$series!="INCOME", ],
NAME="series",X="YYYYMM",Y="value")
PYdf$INCOME<-NULL
head(PYdf)

#Log transformation
```

```
PYdf<-log10(PYdf)

# Generate a plot_ly result
## Note that we've done a log transformation on y-axis
plot_ly(type="scatter",mode="lines")%>%
  add_lines(x=as.yearmon(rownames(PYdf)),text=rownames(PYdf),
  y=PYdf$ICS,name="ICS",line=list(color="powderblue"))%>%
  add_lines(x=as.yearmon(rownames(PYdf)),text=rownames(PYdf),
  y=PYdf$ICC,name="ICC",line=list(color="red"))%>%
  add_lines(x=as.yearmon(rownames(PYdf)),text=rownames(PYdf),
  y=PYdf$GOVT,name="GOVT",line=list(color="green"))%>%
  add_lines(x=as.yearmon(rownames(PYdf)),text=rownames(PYdf),
  y=PYdf$DUR,name="DUR",line=list(color="orange"))%>%
  add_lines(x=as.yearmon(rownames(PYdf)),text=rownames(PYdf),
  y=PYdf$HOM,name="HOM",line=list(color="purple"))%>%
  add_lines(x=as.yearmon(rownames(PYdf)),text=rownames(PYdf),
  y=PYdf$CAR,name="CAR",line=list(color="pink"))%>%
  add_lines(x=as.yearmon(rownames(PYdf)),text=rownames(PYdf),
  y=PYdf$AGE,name="AGE",line=list(color="brown"))%>%
  add_lines(x=as.yearmon(rownames(PYdf)),text=rownames(PYdf),
  y=PYdf$EDUC,name="EDUC",line=list(color="black"))%>%
  layout(title= list(text="Time series for 8 indexes",
  font=list(family = "Times New Roman",size = 16,color = "black" )),
         paper_bgcolor='rgb(255,255,255)', plot_bgcolor='rgb(229,229,229)',
         xaxis = list(title ="Time (monthly)",
                      gridcolor = 'rgb(255,255,255)',
                      showgrid = TRUE,
                      showline = FALSE,
                      showticklabels = TRUE,
                      tickcolor = 'rgb(127,127,127)',
                      ticks = 'outside',
                      zeroline = FALSE),
         yaxis = list(title = "Index Values (log-scale)",
                      gridcolor = 'rgb(255,255,255)',
                      showgrid = TRUE,
                      showline = FALSE,
                      showticklabels = TRUE,
                      tickcolor = 'rgb(127,127,127)',
                      ticks = 'outside',
                      zeroline = FALSE))
```

GTSplot *'GTSplot'*

Description

General Plotly method only working on time series data

Usage

```
GTSplot(tsdata, NEWtitle = "Result", Ylab = "Value", Xlab = "Time",
  Unit = NULL, ts_name = NULL, title_size = 10, COLO = NULL)
```

Arguments

tsdata	A vector or a data frame contains information of time series dataset(i.e. created by ts function)
NEWtitle	title for this plot
Ylab	label of Y axis
Xlab	label of X axis
Unit	the unit of time for the time series data
ts_name	a vector contains names for each time line
title_size	size of the title
COLO	a vector contains colors for each time line

Details

The function TSplot is based on package plotly. It applies plot_ly function to create interactive plot for time series data(i.e data generated by function ts).

Value

a plot result created by plot_ly() function

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
require(forecast)
require(dcemriS4)
require(plotly)

## In the "fMRI" chapter, we have a 4-dimension dataset
## with x,y,z and time dimension (dataset "fMRIVolume").
## So we can settle x,y,and z to determine a vector of time series data.

# You could find the raw "fMRIVolume" dataset on the SOCR website
fMRIURL <- "http://socr.umich.edu/HTML5/BrainViewer/data/fMRI_FilteredData_4D.nii.gz"
fMRIFile <- file.path(tempdir(), "fMRI_FilteredData_4D.nii.gz")
download.file(fMRIURL, dest=fMRIFile, quiet=TRUE)
fMRIVolume <- readNIfTI(fMRIFile, reorient=FALSE)


# Load three time series data(with a wrong format)
xA_fMRI_1D_x20_y20_z11 <- fMRIVolume[20, 20, 11, ]
```

```
xB_fMRI_1D_x30_y30_z13 <- fMRIVolume[30, 30, 13, ]
xC_fMRI_1D_x40_y40_z12 <- fMRIVolume[40, 40, 12, ]

# Change this to time series data
TS1<-ts(xA_fMRI_1D_x20_y20_z11,start=0,frequency =1/3)
TS2<-ts(xB_fMRI_1D_x30_y30_z13,start=0,frequency =1/3)
TS3<-ts(xC_fMRI_1D_x40_y40_z12,start=0,frequency =1/3)

# Package them into a data frame
TSDF<-data.frame(TS1,TS2,TS3)

# Using this function to create plot
GTSplot(TSDF,Xlab="Time(second)",Unit="sec",ts_name=c("xA_fMRI_1D_x20_y20_z11",
"xB_fMRI_1D_x30_y30_z13","xC_fMRI_1D_x40_y40_z12"),
              COLO=c("green","red","blue"))
```

`MCSI_Data_monthAvg_melt`

MCSI_Data_monthAvg_melt

Description

monthly data working on example of `GtoP_trans`. It is a typical dataset that can work on `ggplot2` but not on `plotly`

Usage

```
MCSI_Data_monthAvg_melt
```

Format

dataset with 4428 rows and 3 columns

YYYYMM Indicating year, month and day

series Indicating different features

value value of different feature under a certain time

`MCSI_Data_monthAvg_ts_Y`

MCSI_Data_monthAvg_ts_Y

Description

time series data working on function `TSplot`

Usage

```
MCSI_Data_monthAvg_ts_Y
```

Format

a vector kind dataset created by function `ts()`

`MCSI_Data_monthAvg_ts_Y_test` *MCSI_Data_monthAvg_ts_Y_test*

Description

time series data working on function `TSplot_gen`

Usage

```
MCSI_Data_monthAvg_ts_Y_test
```

Format

a vector kind dataset created by function `ts()`

`modArima_train` *modArima_train*

Description

fitted ARIMA(X) model result. Generated by function `auto.arima()`

Usage

```
modArima_train
```

Format

a list result generated by function `auto.arima()`

TSplot *'TSplot'*

Description

Plotly method working on time series analysis. Work only for month & year dataset(i.e. dataset that can satisfy the format required by function `as.yearmon` from package `zoo`)

Usage

```
TSplot(origin_t, ARIMAmodel, XREG = NULL, NEWtitle = "Result",
  Ylab = "Value", Xlab = "Time(Month/Year)",
  ts_original = "original time series",
  ts_forecast = "forecasted time series", title_size = 10)
```

Arguments

`origin_t`	Number of periods of original time series data you wish to include in the plot write all if all periods should be included
`ARIMAmodel`	ARIMA model created by function `auto.arima()`
`XREG`	if using ARIMAX model, put in the regularized X matrix
`NEWtitle`	title for this plot
`Ylab`	label of Y axis
`Xlab`	label of X axis
`ts_original`	label for original time series line
`ts_forecast`	label for forecasted time series line
`title_size`	size of the title

Details

The function `TSplot` is based on package `plotly`. It applies `plot_ly` function to create interactive plot for time-series analysis result. It requires a fitted model by function `auto.arima`. If you are fitting an ARIMA model with external regressors (i.e. `Xreg`), then you must put inside the external regressors again.

Value

a plot result created by plot_ly() function

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
require(forecast)
require(zoo)
require(plotly)

# Creating time series data
MCSI_Data_monthAvg_ts_Y <- ts(Y, start=c(1978,1), end=c(2018, 12), frequency = 12)

# Applying ARIMAX model
modArima <- auto.arima(MCSI_Data_monthAvg_ts_Y, xreg=X)

# Creating plot_ly results
## 48 means that there will be 48 periods from the original
## time series dataset that is included in the plot result.
## You could also change this to "all" to see all original dataset in a single plot.
TSplot(48,modArima,X_new,title_size = 8,ts_original = "Original time series",
ts_forecast = "Predicted time series")
```

TSplot_gen *'TSplot_gen'*

Description

Plotly method working on time series analysis. This is a more general function that can work on any fitted ARIMA(X) model with any time format. Also, a list of time series data can be applied into this function so that new time lines can be directly drawn in the plot without referring to the result of ADDline function.

Usage

```
TSplot_gen(origin_t, ARIMAmodel, XREG = NULL, NEWtitle = "Result",
  Ylab = "Value", Xlab = "Time", plot_labels = NULL,
  ts_original = "original time series",
  ts_forecast = "forecasted time series", title_size = 10,
  ts_list = "empty", ts_labels = NULL, ts_names = NULL,
  COLO = NULL)
```

Arguments

origin_t	Number of periods of original time series data you wish to include in the plot write all if all periods should be included
ARIMAmodel	ARIMA model created by function auto.arima()
XREG	if using ARIMAX model, put in the regularized X matrix
NEWtitle	title for this plot
Ylab	label of Y axis
Xlab	label of X axis

plot_labels	To include a specific labels for each time point of all training time series data and predicted ARIMA(X) result. A vector should be applied to this parameter that has the same length of chosen periods of training time series data (i.e. parameter origin_t) along with predicted time series periods.
ts_original	label for original time series line
ts_forecast	label for forecasted time series line
title_size	size of the title
ts_list	applying this function can help you draw more time lines into the original plot. A list should be applied to this parameter which contains all extra time series data that you wish to draw on the original plot. Each element on this list should be created by function ts()
ts_labels	when drawing extra time lines with parameter ts_list. You could create specific labels for each time points. A list with the same shape of list in ts_list should be applied. Each element in this list should contain time labels corresponding with the list in ts_list
ts_names	Creating labels for each extra time lines you draw. Labels will appear on the legend of the plot.
COLO	Specifying colors for each new lines that is drawn.

Details

The function TSplot_gen is based on package plotly. It applies plot_ly function to create interactive plot for time-series analysis result. It requires a fitted model by function auto.arima. If you are fitting an ARIMA model with external regressors (i.e. Xreg), then you must put inside the external regressors again.

Value

a plot result created by plot_ly() function

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
# Creating time labels
require(zoo)
require(forecast)
require(plotly)

tl1<-as.yearmon(time(modArima_train$x))[(length(modArima_train$x)-48+1):length(modArima_train$x)]
tl2<-as.yearmon(time(forecast(modArima_train,xreg = as.matrix(X_test))$mean))
tl<-as.character(c(tl1,tl2))
# Creating list and other information for new lines
TSlist<-list(MCSI_Data_monthAvg_ts_Y_test)
TSlabel<-list(as.character(as.yearmon(time(TSlist[[1]]))))
TSname<-c("Original result")
```

```
# Put them into related parameters
TSplot_gen(48,modArima_train,as.matrix(X_test),title_size = 8,ts_original = "Original time series",
        ts_forecast = "Predicted time series",plot_labels = tl, #labels of original plot
        ts_list = TSlist,ts_names = TSname,ts_labels = TSlabel,COLO = "black")
```

X *X*

Description

an external regressor working on function `auto.arima()`. It changes a regular ARIMA model into ARIMAX model

Usage

```
X
```

Format

a matrix serving as the external regressor

---

xA_fMRI_1D_x20_y20_z11 *xA_fMRI_1D_x20_y20_z11*

Description

a vector generated from a 4 dimensions dataset. The original dataset is a 4D dataset containing a 3 dimensions space and a time dimension. this vector is generated by fixing space dimensions to get different value of different time. Fixed (x,y,z) is (20,20,11).

Usage

```
xA_fMRI_1D_x20_y20_z11
```

Format

a vector contains a time series information

xB_fMRI_1D_x30_y30_z13 *xB_fMRI_1D_x30_y30_z13*

Description

a vector generated from a 4 dimensions dataset. The original dataset is a 4D dataset containing a 3 dimensions space and a time dimension. this vector is generated by fixing space dimensions to get different value of different time. Fixed (x,y,z) is (30,30,13).

Usage

```
xB_fMRI_1D_x30_y30_z13
```

Format

a vector contains a time series information

xC_fMRI_1D_x40_y40_z12 *xC_fMRI_1D_x40_y40_z12*

Description

a vector generated from a 4 dimensions dataset. The original dataset is a 4D dataset containing a 3 dimensions space and a time dimension. this vector is generated by fixing space dimensions to get different value of different time. Fixed (x,y,z) is (40,40,12).

Usage

```
xC_fMRI_1D_x40_y40_z12
```

Format

a vector contains a time series information

X_new *X_new*

Description

an external regressor working on function `auto.arima()`. It changes a regular ARIMA model into ARIMAX model

Usage

```
X_new
```

Format

a matrix serving as the external regressor

X_test *X_test*

Description

an external regressor working on function `auto.arima()`. It changes a regular ARIMA model into ARIMAX model

Usage

```
X_test
```

Format

a data frame serving as the external regressor

Y *Y*

Description

information of a time series dataset.

Usage

```
Y
```

Format

a vector

Index

∗Topic **ARIMA(X)**
modArima_train, 8
∗Topic **datasets**
MCSI_Data_monthAvg_melt, 7
∗Topic **dataset**
MCSI_Data_monthAvg_ts_Y, 7
MCSI_Data_monthAvg_ts_Y_test, 8
∗Topic **data**
X_test, 14
∗Topic **frame**
X_test, 14
∗Topic **matrix**
X, 12
X_new, 14
∗Topic **model**
modArima_train, 8
∗Topic **result**
modArima_train, 8
∗Topic **time-series**
MCSI_Data_monthAvg_ts_Y, 7
MCSI_Data_monthAvg_ts_Y_test, 8
∗Topic **vector**
xA_fMRI_1D_x20_y20_z11, 12
xB_fMRI_1D_x30_y30_z13, 13
xC_fMRI_1D_x40_y40_z12, 13
Y, 14

ADDline, 2

GtoP_trans, 3
GTSplot, 5

MCSI_Data_monthAvg_melt, 7
MCSI_Data_monthAvg_ts_Y, 7
MCSI_Data_monthAvg_ts_Y_test, 8
modArima_train, 8

TSplot, 9
TSplot_gen, 10

X, 12
X_new, 14
X_test, 14
xA_fMRI_1D_x20_y20_z11, 12
xB_fMRI_1D_x30_y30_z13, 13
xC_fMRI_1D_x40_y40_z12, 13

Y, 14