Package 'TSstudio'

February 7, 2018

Type Package

Title Tools for Time Series Analysis and Forecasting

Version 0.1.1

Author Rami Krispin

Maintainer Rami Krispin <rami.krispin@gmail.com>

Description Provides a set of interactive visualization tools for time series analysis supporting ts, mts, zoo and xts objects. That includes visualization functions for forecasting model performance (forecasted vs. actual), time series interactive plots (single and multiple series) and seasonality plots.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.0.2)

Imports reshape2 (>= 1.4.2), lubridate (>= 1.6.0), plotly (>= 4.7.1), magrittr (>= 1.5), xts (>= 0.10-1), zoo (>= 1.8-0), forecast (>= 8.2)

Suggests knitr, rmarkdown, devtools, quantmod, DT

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-02-07 21:06:05 UTC

R topics documented:

`check_res` *Visualization of the Residuals of a Time Series Model*

Description

Provides a visualization of the residuals of a time series model. That includes a time series plot of the residuals, and the plots of the autocorrelation function (acf) and histogram of the residuals

Usage

```
check_res(ts.model, lag.max = 36)
```

Arguments

`ts.model`	A time series model (or forecasted) object, support any model from the forecast package with a residuals output
`lag.max`	The maximum number of lags to display in the residuals' autocorrelation function plot

Examples

```
## Not run:
library(forecast)
data(USgas)

# Create a model
fit <- auto.arima(USgas, lambda = BoxCox.lambda(train))

# Check the residuals of the model
check_res(fit)
```

```
## End(Not run)
```

`EURO_Brent` *Crude Oil Prices: Brent - Europe*

Description

Crude Oil Prices: Brent - Europe: 1987 - 2017. Units: Dollars per Barrel

Usage

```
EURO_Brent
```

Format

Time series data - 'zoo'

Source

U.S. Energy Information Administration, Crude Oil Prices: Brent - Europe [MCOILBRENTEU], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/MCOILBRENTEU, January 8, 2018.

Examples

```
ts.plot_ly(EURO_Brent)
seasonal_ly(EURO_Brent)
```

`Michigan_CS` *University of Michigan Consumer Survey, Index of Consumer Sentiment*

Description

University of Michigan Consumer Survey, Index of Consumer Sentiment: 1980 - 2017. Units: Index 1966:Q1=100

Usage

```
Michigan_CS
```

Format

Time series data - 'xts' object

Source

University of Michigan, University of Michigan: Consumer Sentiment

Examples

```
ts.plot_ly(Michigan_CS)
seasonal_ly(Michigan_CS)
```

`res_hist` *Histogram Plot of the Residuals Values*

Description

Histogram plot of the residuals values

Usage

```
res_hist(forecast.obj)
```

Arguments

`forecast.obj`	A fitted or forecasted object (of the forecast package) with residuals output

Examples

```
## Not run:
library(forecast)
data(USgas)

# Set the horizon of the forecast
h <- 12

# split to training/testing partition
split_ts <- ts_split(USgas, sample.out  = h)
train <- split_ts$train
test <- split_ts$test

# Create forecast object
fc <- forecast(auto.arima(train, lambda = BoxCox.lambda(train)), h = h)

# Plot the fitted and forecasted vs the actual values
res_hist(forecast.obj = fc)

## End(Not run)
```

test_forecast *Visualize of the Fitted and the Forecasted vs the Actual Values*

Description

Visualize the fitted values of the training set and the forecast values of the testing set against the actual values of the series

Usage

```
test_forecast(actual, forecast.obj, train = NULL, test, Ygrid = FALSE,
  Xgrid = FALSE, hover = TRUE)
```

Arguments

actual	The full time series object (support "ts", "zoo" and "xts" formats)
forecast.obj	The forecast output of the training set with horizon align to the length of the testing (support forecasted objects from the "forecast" package)
train	Training partition, a subset of the first n observation in the series (not requiredthed)
test	The testing (hold-out) partition
Ygrid	Logic,show the Y axis grid if set to TRUE
Xgrid	Logic,show the X axis grid if set to TRUE
hover	If TRUE add tooltip with information about the model accuracy

Examples

```
## Not run:
library(forecast)
data(USgas)

# Set the horizon of the forecast
h <- 12

# split to training/testing partition
split_ts <- ts_split(USgas, sample.out  = h)
train <- split_ts$train
test <- split_ts$test

# Create forecast object
fc <- forecast(auto.arima(train, lambda = BoxCox.lambda(train)), h = h)

# Plot the fitted and forecasted vs the actual values
test_forecast(actual = USgas, forecast.obj = fc, test = test)

## End(Not run)
```

ts_acf *A Visualization Function of the ACF Estimation*

Description

A Visualization Function of the ACF Estimation

Usage

```
ts_acf(ts.obj, lag.max = NULL, ci = 0.95, color = NULL)
```

Arguments

ts.obj	a univariate or multivariate time series object of class "ts", "mts", "zoo" or "xts"
lag.max	maximum lag at which to calculate the acf. Default is 10*log10(N/m) where N is the number of observations and m the number of series. Will be automatically limited to one less than the number of observations in the series.
ci	the significant level of the estimation - a numeric value between 0 and 1, default is set for 0.95
color	The color of the plot, support both name and expression

Examples

```
data(USgas)

ts_acf(USgas, lag.max = 60)
```

ts_decompose *Visualization of the Decompose of a Time Series Object*

Description

Interactive visualization the trend, seasonal and random components of a time series based on the decompose function from the stats package.

Usage

```
ts_decompose(ts.obj, type = "additive", showline = TRUE)
```

Arguments

ts.obj	a univariate time series object of a class "ts", "zoo" or "xts"
type	Set the type of the seasonal component, can be set to either "additive", "multiplicative" or "both" to compare between the first two options (default set to “additive”)
showline	Logic, add a separation line between each of the plot components (default set to TRUE)

Examples

```
# Defualt decompose plot
ts_decompose(AirPassengers)

# Remove the sepration lines between the plot components
ts_decompose(AirPassengers, showline = FALSE)

# Plot side by side a decompose of additive and multiplicative series
ts_decompose(AirPassengers, type = "both")
```

`ts_heatmap` *Heatmap Plot for Time Series*

Description

Heatmap plot for time series object by it periodicity (currently support only monthly and quarterly frequency)

Usage

```
ts_heatmap(ts.obj)
```

Arguments

`ts.obj`	a univariate time series object of a class "ts", "zoo" or "xts" (support only series with either monthly or quarterly frequency)

Examples

```
data(USgas)
ts_heatmap(USgas)
```

`ts_lags` *Time Series Lag Visualization*

Description

Visualization of series with its lags, can be used to identify a correlation between the series and it lags

Usage

```
ts_lags(ts.obj, lag.max = 12, Xtitle = FALSE, Ytitle = TRUE,
  margin = 0.02, Xshare = TRUE, Yshare = TRUE, n_row = 3)
```

Arguments

ts.obj	A univariate time series object of a class "ts", "zoo" or "xts" (support only series with either monthly or quarterly frequency)
lag.max	An integer, number of lags to plot
Xtitle	Plotly parameter, should x-axis titles be retained?
Ytitle	Plotly parameter, should y-axis titles be retained?
margin	Plotly parameter, either a single value or four values (all between 0 and 1). If four values are provided, the first is used as the left margin, the second is used as the right margin, the third is used as the top margin, and the fourth is used as the bottom margin. If a single value is provided, it will be used as all four margins.
Xshare	Plotly parameter, should the x-axis be shared amongst the subplots?
Yshare	Plotly parameter, should the y-axis be shared amongst the subplots?
n_row	An integer, define the number of plots per row

Examples

```
data(USgas)

ts_lags(USgas)
```

---

ts_pacf *A Visualization Function of the PACF Estimation*

Description

A Visualization Function of the PACF Estimation

Usage

```
ts_pacf(ts.obj, lag.max = NULL, ci = 0.95, color = NULL)
```

Arguments

ts.obj	a univariate or multivariate time series object of class "ts", "mts", "zoo" or "xts"
lag.max	maximum lag at which to calculate the acf. Default is 10*log10(N/m) where N is the number of observations and m the number of series. Will be automatically limited to one less than the number of observations in the series.
ci	the significant level of the estimation - a numeric value between 0 and 1, default is set for 0.95
color	The color of the plot, support both name and expression

Examples

```
data(USgas)

ts_pacf(USgas, lag.max = 60)
```

ts_plot *Plotting Time Series Objects*

Description

Visualization functions for time series object

Usage

```
ts_plot(ts.obj, line.mode = "lines", width = 2, dash = NULL,
  color = NULL, slider = FALSE, type = "multiple", Xtitle = NULL,
  Ytitle = NULL, title = NULL, Xgrid = FALSE, Ygrid = FALSE)
```

Arguments

ts.obj	A univariate or multivariate time series object of class "ts", "mts", "zoo" or "xts"
line.mode	A plotly argument, define the plot type, c("lines", "lines+markers", "markers")
width	The plot width, default is set to 1 (an integer)
dash	A plotly argument, define the line style, c(NULL, "dot", "dash")
color	The color of the plot, support both name and expression
slider	Logic, add slider to modify the time axis (default set to FALSE)
type	Applicable for multiple time series object, plot on a separate plot or all together c("single, "multiple)
Xtitle	Set the X axis title, default set to NULL
Ytitle	Set the Y axis title, default set to NULL
title	Set the plot title, default set to NULL
Xgrid	Logic,show the X axis grid if set to TRUE
Ygrid	Logic,show the Y axis grid if set to TRUE

Examples

```
data(USVSales)
ts_plot(USVSales)

# adding slider
ts_plot(USVSales, slider = TRUE)
```

ts_polar *Polor Plot for Time Series Object*

Description

Polor plot for time series object (ts, zoo, xts), currently support only monthly and quarterly frequency

Usage

```
ts_polar(ts.obj, title = NULL, width = 600, height = 600, left = 25,
  right = 25, top = 25, bottom = 25)
```

Arguments

ts.obj	A univariate time series object of a class "ts", "zoo" or "xts" (support only series with either monthly or quarterly frequency)
title	Add a title for the plot, default set to NULL
width	The widht of the plot in pixels, default set to 600
height	The height of the plot pixels, default set to 600
left	Set the left margin of the plot in pixels, default set to 25
right	Set the right margin of the plot in pixels, default set to 25
top	Set the top margin of the plot in pixels, default set to 25
bottom	Set the bottom margin of the plot in pixels, default set to 25

Examples

```
data(USgas)
ts_polar(USgas)
```

ts_reshape *Transform Time Series Object to Data Frame Format*

Description

Transform time series object into data frame format

Usage

```
ts_reshape(ts.obj, type = "wide")
```

Arguments

ts.obj	a univariate time series object of a class "ts", "zoo" or "xts" (support only series with either monthly or quarterly frequency)
type	The reshape type - "wide" set the years as the columns and the cycle units (months or quarter) as the rows, or "long" split the time object to year, cycle unit and value

Examples

```
data(USgas)
USgas_df <- ts_reshape(USgas)
```

---

ts_seasonal *Seasonality Visualization of Time Series Object*

Description

Visualize time series object by it periodicity, currently support only monthly and quarterly frequency

Usage

```
ts_seasonal(ts.obj, type = "normal", Ygrid = FALSE, Xgrid = FALSE)
```

Arguments

ts.obj	a univariate time series object of a class "ts", "zoo" or "xts" (support only series with either monthly or quarterly frequency)
type	The type of the seasonal plot - "normal" to split the series by full cycle units, or "cycle" to split by cycle units, or "box" for box-plot by cycle units, or "all" for all the three plots together
Ygrid	logic,show the Y axis grid if set to TRUE
Xgrid	logic,show the X axis grid if set to TRUE

Examples

```
data(USgas)
ts_seasonal(USgas)

# Seasonal box plot
ts_seasonal(USgas, type = "box")
```

ts_split *Split Time Series Object for Training and Testing Partitions*

Description

Split a time series object into training and testing partitions

Usage

```
ts_split(ts.obj, sample.out = NULL)
```

Arguments

ts.obj	A univariate time series object of a class "ts"
sample.out	An integer, set the number of periods of the testing or sample out partition, defualt set for 30 percent of the lenght of the series

Examples

```
# Split the AirPassengers into training and testing partitions

# Set the last 12 months as a testing partition

# and the rest as a training partition

data(AirPassengers)

split_Air <- ts_split(ts.obj = AirPassengers, sample.out = 12)

training <- split_Air$train
testing <- split_Air$test

length(AirPassengers)

length(training)

length(testing)
```

ts_surface *3D Surface Plot for Time Series*

Description

3D surface plot for time series object by it periodicity (currently support only monthly and quarterly frequency)

Usage

```
ts_surface(ts.obj)
```

Arguments

ts.obj	a univariate time series object of a class "ts", "zoo" or "xts" (support only series with either monthly or quarterly frequency)

Examples

```
ts_surface(USgas)
```

USgas *US monthly natural gas consumption*

Description

US monthly natural gas consumption: 2000 - 2017. Units: Billion Cubic Feet

Usage

```
USgas
```

Format

Time series data - 'ts' object

Source

U.S. Bureau of Transportation Statistics, Natural Gas Consumption [NATURALGAS], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/NATURALGAS, January 7, 2018.

Examples

```
ts.plot_ly(USgas)
seasonal_ly(USgas)
```

USUnRate *US Monthly Civilian Unemployment Rate*

Description

US monthly civilian unemployment rate: 1948 - 2017. Units: Percent

Usage

```
USUnRate
```

Format

Time series data - 'ts' object

Source

U.S. Bureau of Labor Statistics, Civilian Unemployment Rate [UNRATENSA], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/UNRATENSA, January 6, 2018.

Examples

```
ts.plot_ly(USUnRate)
seasonal_ly(USUnRate)
```

USVSales *US Monthly Total Vehicle Sales*

Description

US monthly total vehicle sales: 1976 - 2017. Units: Thousands of units

Usage

```
USVSales
```

Format

Time series data - 'ts' object

Source

U.S. Bureau of Economic Analysis, Total Vehicle Sales [TOTALNSA], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/TOTALNSA, January 7, 2018.

Examples

```
ts.plot_ly(USVSales)
seasonal_ly(USVSales)
```

---

xts_to_ts *Converting 'xts' object to 'ts' object*

Description

Converting 'xts' object to 'ts' object

Usage

```
xts_to_ts(xts.obj)
```

Arguments

xts.obj	a univariate 'xts' object

Examples

```
data("Michigan_CS", package = "TSstudio")
class(Michigan_CS)
ts.plot_ly(Michigan_CS)
Michigan_CS_ts <- xts_to_ts(Michigan_CS)
ts.plot_ly(Michigan_CS_ts)
```

---

zoo_to_ts *Converting 'zoo' object to 'ts' object*

Description

Converting 'zoo' object to 'ts' object

Usage

```
zoo_to_ts(zoo.obj)
```

Arguments

zoo.obj	a univariate 'zoo' object

Examples

```
data("EURO_Brent", package = "TSstudio")
class(EURO_Brent)
ts.plot_ly(EURO_Brent)
EURO_Brent_ts <- zoo_to_ts(EURO_Brent)
class(EURO_Brent_ts)
ts.plot_ly(EURO_Brent_ts)
```

Index