Package ‘OOS’

October 12, 2022

Title Out-of-Sample Time Series Forecasting

Version 1.0.0

Description A comprehensive and cohesive API for the out-of-sample forecasting workflow: data preparation, forecasting - including both traditional econometric time series models and modern machine learning techniques - forecast combination, model and error analysis, and forecast visualization.

License GPL-3


BugReports https://github.com/tylerJPike/OOS/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

VignetteBuilder knitr

Depends R (>= 4.0.0)

Imports caret, dplyr, forecast, furrr, future, ggplot2, glmnet, imputeTS, lmtest, lubridate, magrittr, purrrr, sandwich, stats, tidyr, vars, xts, zoo

Suggests knitr, testthat, rmarkdown, quantmod

NeedsCompilation no

Author Tyler J. Pike [aut, cre]

Maintainer Tyler J. Pike <tjpike7@gmail.com>

Repository CRAN

Date/Publication 2021-03-17 13:20:20 UTC

R topics documented:

```
chart_forecast .................................................. 2
chart_forecast_error ........................................... 3
```

**chart_forecast**

**Description**

Chart forecasts

**Usage**

```r
chart_forecast(Data, Title, Ylab, Freq, zeroline = FALSE)
```

**Arguments**

- **Data**
  - data.frame: oos.forecast object
- **Title**
  - string: chart title
- **Ylab**
  - string: y-axis label
- **Freq**
  - string: frequency (acts as sub-title)
- **zeroline**
  - boolean: if TRUE then add a horizontal line at zero

**Value**

ggplot2 chart
Examples

# simple time series
A = c(1:100) + rnorm(100)
date = seq.Date(from = as.Date('2000-01-01'), by = 'month', length.out = 100)
Data = data.frame(date = date, A)

# run forecast_univariate
forecast.uni =
  forecast_univariate(
    Data = Data,
    forecast.dates = tail(Data$date,10),
    method = c('naive','auto.arima', 'ets'),
    horizon = 1,
    recursive = FALSE,
    freq = 'month')

forecasts =
dplyr::left_join(
  forecast.uni,
  data.frame(date, observed = A),
  by = 'date'
)

# chart forecasts
chart.forecast =
  chart_forecast(
    forecasts,
    Title = 'test',
    Ylab = 'Index',
    Freq = 'Monthly',
    zeroline = TRUE)

---

chart_forecast_error  Chart forecast errors

Description

Chart forecast errors

Usage

chart_forecast_error(Data, Title, Ylab, Freq, zeroline = FALSE)
Arguments

- **Data**
  - data.frame: oos.forecast object
- **Title**
  - string: chart title
- **Ylab**
  - string: y-axis label
- **Freq**
  - string: frequency (acts as sub-title)
- **zeroline**
  - boolean: if TRUE then add a horizontal line at zero

Value

- ggplot2 chart

Examples

```r
# simple time series
A = c(1:100) + rnorm(100)
date = seq.Date(from = as.Date('2000-01-01'), by = 'month', length.out = 100)
Data = data.frame(date = date, A)

# run forecast_univariate
forecast.uni =
  forecast_univariate(
    Data = Data,
    forecast.dates = tail(Data$date,10),
    method = c('naive','auto.arima', 'ets'),
    horizon = 1,
    recursive = FALSE,
    freq = 'month')

forecasts =
  dplyr::left_join(
    forecast.uni,
    data.frame(date, observed = A),
    by = 'date'
  )

# chart forecast errors
chart.errors =
  chart_forecast_error(
    forecasts,
    Title = 'test',
    Ylab = 'Index',
    Freq = 'Monthly',
    zeroline = TRUE)
```
**data_impute**  
*Impute missing values*

**Description**
A function to impute missing values. Is used as a data preparation helper function and is called internally by `forecast_univariate`, `forecast_multivariate`, and `forecast_combine`.

**Usage**
```r
data_impute(Data, method = "kalman", variables = NULL, verbose = FALSE)
```

**Arguments**
- **Data**  
  data.frame: data frame of target variable, exogenous variables, and observed date (named 'date')
- **method**  
  string: select which method to use from the `imputeTS` package: 'interpolation', 'kalman', 'locf', 'ma', 'mean', 'random', 'remove', 'replace', 'seadec', 'seasplit'
- **variables**  
  string: vector of variables to standardize, default is all but 'date' column
- **verbose**  
  boolean: show start-up status of `impute.missing.routine`

**Value**
data.frame with missing data imputed

---

**data_outliers**  
*Clean outliers*

**Description**
A function to clean outliers. Is used as a data preparation helper function and is called internally by `forecast_univariate`, `forecast_multivariate`, and `forecast_combine`.

**Usage**
```r
data_outliers(  
  Data,  
  variables = NULL,  
  w.bounds = c(0.05, 0.95),  
  trim = FALSE,  
  cross_section = FALSE  
)
```
**data_reduction**

**Description**

A function to estimate principal components.

**Usage**

```
data_reduction(Data, variables = NULL, ncomp, standardize = TRUE)
```

**Arguments**

- `Data`  
  data.frame: data frame of target variable, exogenous variables, and observed date (named 'date')
- `variables`  
  string: vector of variables to standardize, default is all but 'date' column
- `ncomp`  
  int: number of factors to create
- `standardize`  
  boolean: normalize variables (mean zero, variance one) before estimating factors

**Value**

data.frame with a date column and one column per forecast method selected
**data_subset**

*Create information set*

**Description**
A function to subset data recursively or with a rolling window to create a valid information set. Is used as a data preparation helper function and is called internally by forecast_univariate, forecast_multivariate, and forecast_combine.

**Usage**

```
data_subset(Data, forecast.date, rolling.window, freq)
```

**Arguments**

- **Data**
  data.frame: data frame of target variable, exogenous variables, and observed date (named 'date')
- **forecast.date**
  date: upper bound of information set
- **rolling.window**
  int: size of rolling window, NA if expanding window is used
- **freq**
  string: time series frequency; day, week, month, quarter, year; only needed for rolling window factors

**Value**

data.frame bounded by the given date range

**forecast_accuracy**

*Calculate forecast accuracy*

**Description**
A function to calculate various loss functions, including MSE, RMSE, MAE, and MAPE.

**Usage**

```
forecast_accuracy(Data)
```

**Arguments**

- **Data**
  data.frame: data frame of forecasts, model names, and dates

**Value**

data.frame of numeric error results
Examples

```r
# simple time series
A = c(1:100) + rnorm(100)
date = seq.Date(from = as.Date('2000-01-01'), by = 'month', length.out = 100)
Data = data.frame(date = date, A)

# run forecast_univariate
forecast.uni =
  forecast_univariate(
    Data = Data,
    forecast.dates = tail(Data$date, 10),
    method = c('naive', 'auto.arima', 'ets'),
    horizon = 1,
    recursive = FALSE,
    freq = 'month')

forecasts =
  dplyr::left_join(
    forecast.uni,
    data.frame(date, observed = A),
    by = 'date')

# forecast accuracy
forecast.accuracy = forecast.accuracy(forecasts)
```

Description

A function to combine forecasts out-of-sample. Methods available include: uniform weights, median forecast, trimmed (winsorized) mean, n-best, ridge regression, lasso regression, elastic net, peLASSO, random forest, tree-based gradient boosting machine, and single-layer neural network. See package website for most up-to-date list of available models.

Usage

```r
forecast_combine(Data,
                  method = "uniform",
                  n.max = NULL,
                  rolling.window = NA,
                  trim = c(0.5, 0.95),
                  ...)```
Arguments

Data   data.frame: data frame of forecasted values to combine, assumes 'date' and 'observed' columns, but 'observed' is not necessary for all methods

method string: the method to use; 'uniform', 'median', 'trimmed.mean', 'n.best', 'peLasso', 'lasso', 'ridge', 'elastic', 'RF', 'GBM', 'NN'

n.max   int: maximum number of forecasts to select in n.best method

rolling.window   int: size of rolling window to evaluate forecast error over, use entire period if NA

trim   numeric: a two element vector with the winsorizing bounds for the trimmed mean method; c(min, max)

burn.in   int: the number of periods to use in the first model estimation

parallel.dates   int: the number of cores available for parallel estimation

Value

data.frame with a row for each combination method and forecasted date

Examples

# simple time series
A = c(1:100) + rnorm(100)
B = c(1:100) + rnorm(100)
C = c(1:100) + rnorm(100)
date = seq.Date(from = as.Date('2000-01-01'), by = 'month', length.out = 100)
Data = data.frame(date = date, A, B, C)

# run forecast_univariate
forecast_multi =
    forecast_multivariate(
        Data = Data,
        target = 'A',
        forecast.dates = tail(Data$date, 5),
        method = c('ols', 'var'),
        horizon = 1,
        freq = 'month')

# include observed valuesd
forecasts =
dplyr::left_join(
    forecast_multi,
    data.frame(date, observed = A),
    by = 'date'  
)
# combine forecasts
combinations =
forecast_combine(
  forecasts,
  method = c('uniform','median','trimmed.mean',
             'n.best','lasso','peLasso'),
  burn.in = 5,
  n.max = 2)

---

**forecast_comparison**  
_Compare forecast accuracy_

**Description**

A function to compare forecasts. Options include: simple forecast error ratios, Diebold-Mariano test, and Clark and West test for nested models

**Usage**

```r
forecast_comparison(
  Data,
  baseline.forecast,
  test = "ER",
  loss = "MSE",
  horizon = NULL
)
```

**Arguments**

- **Data**
  data.frame: data frame of forecasts, model names, and dates
- **baseline.forecast**
  string: column name of baseline (null hypothesis) forecasts
- **test**
  string: which test to use; ER = error ratio, DM = Diebold-Mariano, CM = Clark and West
- **loss**
  string: error loss function to use if creating forecast error ratio
- **horizon**
  int: horizon of forecasts being compared in DM and CW tests

**Value**

numeric test result
**Examples**

```r
# simple time series
A = c(1:100) + rnorm(100)
date = seq.Date(from = as.Date(’2000-01-01’), by = ’month’, length.out = 100)
Data = data.frame(date = date, A)

# run forecast_univariate
forecast.uni =
  forecast_univariate(
    Data = Data,
    forecast.dates = tail(Data$date,10),
    method = c(’naive’,’auto.arima’, ’ets’),
    horizon = 1,
    recursive = FALSE,
    freq = ’month’)

forecasts =
  dplyr::left_join(
    forecast.uni,
    data.frame(date, observed = A),
    by = ’date’
  )

# run ER (MSE)
er.ratio.mse =
  forecast_comparison(
    forecasts,
    baseline.forecast = ’naive’,
    test = ’ER’,
    loss = ’MSE’)
```

**Description**

A function to subset data recursively or with a rolling window to create a valid information set. Is used as a data preparation helper function and is called internally by forecast_univariate, forecast_multivariate, and forecast_combine.

**Usage**

```r
forecast_date(forecast.date, horizon, freq)
```
**forecast_multivariate**

**Arguments**

- `forecast.date`: date forecast was made
- `horizon`: int: periods ahead of forecast
- `freq`: string: time series frequency; day, week, month, quarter, year; only needed for rolling window factors

**Value**

date vector

**Description**

A function to estimate multivariate forecasts out-of-sample. Methods available include: vector auto-regression, linear regression, lasso regression, ridge regression, elastic net, random forest, tree-based gradient boosting machine, and single-layer neural network. See package website for most up-to-date list of available models.

**Usage**

```r
forecast_multivariate(
  Data,
  forecast.dates,
  target,
  horizon,
  method,
  rolling.window = NA,
  freq,
  lag.variables = NULL,
  lag.n = NULL,
  outlier.clean = FALSE,
  outlier.variables = NULL,
  outlier.bounds = c(0.05, 0.95),
  outlier.trim = FALSE,
  outlier.cross_section = FALSE,
  impute.missing = FALSE,
  impute.method = "kalman",
  impute.variables = NULL,
  impute.verbose = FALSE,
  reduce.data = FALSE,
  reduce.variables = NULL,
  reduce.ncomp = NULL,
  reduce.standardize = TRUE,
  parallel.dates = NULL,
)```

return.models = FALSE,
return.data = FALSE
)

**Arguments**

**Data**
- data.frame: data frame of target variable, exogenous variables, and observed date (named 'date'); may alternatively be a ts, xts, or zoo object to forecast

**forecast.dates**
- date: dates forecasts are created

**target**
- string: column name in Data of variable to forecast

**horizon**
- int: number of periods into the future to forecast

**method**
- string: methods to use

**rolling.window**
- int: size of rolling window, NA if expanding window is used

**freq**
- string: time series frequency; day, week, month, quarter, year

**lag.variables**
- string: vector of variables to lag each time step, if lag.n is not null then the default is all non-date variables

**lag.n**
- int: number of lags to create

**outlier.clean**
- boolean: if TRUE then clean outliers

**outlier.variables**
- string: vector of variables to purge of outlier, default is all but 'date' column

**outlier.bounds**
- double: vector of winsorizing minimum and maximum bounds, c(min percentile, max percentile)

**outlier.trim**
- boolean: if TRUE then replace outliers with NA instead of winsorizing bound

**outlier.cross_section**
- boolean: if TRUE then remove outliers based on cross-section (row-wise) instead of historical data (column-wise)

**impute.missing**
- boolean: if TRUE then impute missing values

**impute.method**
- string: select which method to use from the imputeTS package; 'interpolation', 'kalman', 'locf', 'ma', 'mean', 'random', 'remove', 'replace', 'seadec', 'seasplit'

**impute.variables**
- string: vector of variables to impute missing values, default is all numeric columns

**impute.verbose**
- boolean: show start-up status of impute.missing.routine

**reduce.data**
- boolean: if TRUE then reduce dimension

**reduce.variables**
- string: vector of variables to impute missing values, default is all numeric columns

**reduce.ncomp**
- int: number of factors to create

**reduce.standardize**
- boolean: normalize variables (mean zero, variance one) before estimating factors

**parallel.dates**
- int: the number of cores available for parallel estimation

**return.models**
- boolean: if TRUE then return list of models estimated each forecast.date

**return.data**
- boolean: if True then return list of information.set for each forecast.date
Value

data.frame with a row for each forecast by model and forecasted date

Examples

# simple time series
A = c(1:100) + rnorm(100)
B = c(1:100) + rnorm(100)
C = c(1:100) + rnorm(100)
date = seq.Date(from = as.Date('2000-01-01'), by = 'month', length.out = 100)
Data = data.frame(date = date, A, B, C)

# run forecast_univariate
forecast.multi =
  forecast_multivariate(
    Data = Data,
    target = 'A',
    forecast.dates = tail(Data$date,5),
    method = c('ols','var'),
    horizon = 1,
    # information set
    rolling.window = NA,
    freq = 'month',
    # data prep
    lag.n = 4,
    outlier.clean = TRUE,
    impute.missing = TRUE)

forecast_univariate  Forecast with univariate models

Description

A function to estimate univariate forecasts out-of-sample. Methods available include all forecast
methods from the forecast package. See package website for most up-to-date list of available
models.

Usage

forecast_univariate(
  Data,
  forecast.dates,
  methods,
  horizon,
  recursive = TRUE,
forecast_univariate

rolling.window = NA,
freq,
outlier.clean = FALSE,
outlier.variables = NULL,
outlier.bounds = c(0.05, 0.95),
outlier.trim = FALSE,
outlier.cross_section = FALSE,
impute.missing = FALSE,
impute.method = "kalman",
impute.variables = NULL,
impute.verbose = FALSE,
parallel.dates = NULL,
return.models = FALSE,
return.data = FALSE
)

Arguments

Data data.frame: data frame of variable to forecast and a date column; may alternatively be a ts, xts, or zoo object to forecast
forecast.dates date: dates forecasts are created
methods string: models to estimate forecasts
horizon int: number of periods to forecast
recursive boolean: use sequential one-step-ahead forecast if TRUE, use direct projections if FALSE
rolling.window int: size of rolling window, NA if expanding window is used
freq string: time series frequency; day, week, month, quarter, year
outlier.clean boolean: if TRUE then clean outliers
outlier.variables string: vector of variables to purge of outliers, default is all but 'date' column
outlier.bounds double: vector of winsorizing minimum and maximum bounds, c(min percentile, max percentile)
outlier.trim boolean: if TRUE then replace outliers with NA instead of winsorizing bound
outlier.cross_section boolean: if TRUE then remove outliers based on cross-section (row-wise) instead of historical data (column-wise)
impute.missing boolean: if TRUE then impute missing values
impute.method string: select which method to use from the imputeTS package; 'interpolation', 'kalman', 'locl', 'ma', 'mean', 'random', 'remove', 'replace', 'seadec', 'seasplit'
impute.variables string: vector of variables to impute missing values, default is all numeric columns
impute.verbose boolean: show start-up status of impute.missing.routine
parallel.dates int: the number of cores available for parallel estimation
return.models boolean: if TRUE then return list of models estimated each forecast.date
return.data boolean: if True then return list of information.set for each forecast.date
Value

data.frame with a row for each forecast by model and forecasted date

Examples

# simple time series
A = c(1:100) + rnorm(100)
date = seq.Date(from = as.Date('2000-01-01'), by = 'month', length.out = 100)
Data = data.frame(date = date, A)

# estimate univariate forecasts
forecast.uni =
    forecast_univariate(
        Data = Data,
        forecast.dates = tail(Data$date, 5),
        method = c('naive', 'auto.arima', 'ets'),
        horizon = 1,
        recursive = FALSE,
        # information set
        rolling.window = NA,
        freq = 'month',
        # data prep
        outlier.clean = TRUE,
        impute.missing = TRUE)
instantiate.forecast_combinations.control_panel

Create interface to control forecast_combine model estimation

Description
A function to create the forecast combination technique arguments list for user manipulation.

Usage
instantiate.forecast_combinations.control_panel(covariates = NULL)

Arguments
covariates int: the number of features that will go into the model

Value
forecast_combinations.control_panel

instantiate.forecast_multivariate.ml.control_panel

Create interface to control forecast_multivariate ML estimation

Description
A function to create the multivariate forecast methods arguments list for user manipulation.

Usage
instantiate.forecast_multivariate.ml.control_panel(
covariates = NULL,
rolling.window = NULL,
horizon = NULL
)

Arguments
covariates int: the number of features that will go into the model
rolling.window int: size of rolling window, NA if expanding window is used
horizon int: number of periods into the future to forecast

Value
forecast_multivariate.ml.control_panel
**instantiate.forecast_multivariate.var.control_panel**

*Create interface to control forecast_multivariate VAR estimation*

**Description**

A function to create the multivariate forecast methods arguments list for user manipulation.

**Usage**

    instantiate.forecast_multivariate.var.control_panel()

**Value**

    forecast_multivariate.var.control_panel

---

**instantiate.forecast_univariate.control_panel**

*Create interface to control forecast_univariate model estimation*

**Description**

A function to create the univariate forecast method arguments list for user manipulation.

**Usage**

    instantiate.forecast_univariate.control_panel()

**Value**

    forecast_univariate.control_panel
**loss_function**

*Calculate error via loss functions*

**Description**

A function to calculate various error loss functions. Options include: MSE, RMSE, MAE, and MAPE. The default is MSE loss.

**Usage**

```r
loss_function(forecast, observed, metric = "MSE")
```

**Arguments**

- **forecast**: numeric: vector of forecasted values
- **observed**: numeric: vector of observed values
- **metric**: string: loss function

**Value**

numeric test result

---

**n.lag**

*Create n lags*

**Description**

A function to create 1 through n lags of a set of variables. Is used as a data preparation helper function and is called internally by forecast_univariate, forecast_multivariate, and forecast_combine.

**Usage**

```r
n.lag(Data, lags, variables = NULL)
```

**Arguments**

- **Data**: data.frame: data frame of variables to lag and a 'date' column
- **lags**: int: number of lags to create
- **variables**: string: vector of variable names to lag, default is all non-date variables

**Value**

data.frame
NBest

Select N-best forecasts

Description
A function to subset the n-best forecasts; assumes column named observed.

Usage
NBest(forecasts, n.max, window = NA)

Arguments
- forecasts: data.frame: a data frame of forecasts to combine, assumes one column named "observed"
- n.max: int: maximum number of forecasts to select
- window: int: size of rolling window to evaluate forecast error over, use entire period if NA

Value
data.frame with n columns of the historically best forecasts

standardize

Standardize variables (mean 0, variance 1)

Description
Standardize variables (mean 0, variance 1)

Usage
standardize(X)

Arguments
- X: numeric: vector to be standardized

Value
numeric vector of standardized values
winsorize

Winsorize or trim variables

Description

Winsorize or trim variables

Usage

winsorize(X, bounds, trim = FALSE)

Arguments

- **X**: numeric: vector to be winsorized or trimmed
- **bounds**: double: vector of winsorizing minimum and maximum bounds, c(min percentile, max percentile)
- **trim**: boolean: if TRUE then replace outliers with NA instead of winsorizing bound

Value

numeric vector of winsorized or trimmed values
Index

chart_forecast, 2
chart_forecast_error, 3
data_impute, 5
data_outliers, 5
data_reduction, 6
data_subset, 7
forecast_accuracy, 7
forecast_combine, 8
forecast_comparison, 10
forecast_date, 11
forecast_multivariate, 12
forecast_univariate, 14

instantiate.data_impute.control_panel, 16
instantiate.forecast_combinations.control_panel, 17
instantiate.forecast_multivariate.ml.control_panel, 17
instantiate.forecast_multivariate.var.control_panel, 18
instantiate.forecast_univariate.control_panel, 18

loss_function, 19

n.lag, 19
NBest, 20

standardize, 20

winsorize, 21