

Package: rjd3nowcasting (via r-universe)

October 10, 2024

Type Package

Title Nowcasting with 'JDemetra+ 3.0'

Version 2.0.3

Description Interface around 'JDemetra+ 3.x'
(<<https://github.com/jdemetra/jdplus-nowcasting>>), TSACE project. It defines and estimates Dynamic Factor Models with the purpose of Nowcasting. News analysis is included in this second version.

Depends R (>= 4.1.0)

Imports rJava (>= 1.0-6), rjd3toolkit (>= 3.2.2)

Remotes github::rjdverse/rjd3toolkit@*release

SystemRequirements Java (>= 17)

License EUPL

URL <https://github.com/rjdverse/rjd3nowcasting>,
<https://rjdverse.github.io/rjd3nowcasting/>

BugReports <https://github.com/rjdverse/rjd3nowcasting/issues>

Encoding UTF-8

Collate 'estimation.R' 'news.R' 'results.R' 'zzz.R'

RoxygenNote 7.3.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

Repository <https://rjdverse.r-universe.dev>

RemoteUrl <https://github.com/rjdverse/rjd3nowcasting>

RemoteRef v2.0.3

RemoteSha ba745713fb2df3505c8d363e4d337ab407c07bed

Contents

create_model	2
estimate_em	4
estimate_ml	6
estimate_pca	8
get_forecasts	9
get_news	10
get_results	11
plot.JD3_DfmForecasts	12
plot.JD3_DfmNews	12
print.JD3_DfmEstimates	13
print.JD3_DfmForecasts	13
print.JD3_DfmNews	14
print.JD3_DfmResults	14
summary.JD3_DfmNews	15
Index	16

create_model	<i>Create Dynamic Factor Model</i>
--------------	------------------------------------

Description

Create Dynamic Factor Model

Usage

```
create_model(
  nfactors,
  nlags,
  factors_type,
  factors_loading,
  var_init = c("Unconditional", "Zero"),
  var_coefficients = NULL,
  var_errors_variance = NULL,
  measurement_coefficients = NULL,
  measurement_errors_variance = NULL
)
```

Arguments

nfactors	Integer. Number of factors.
nlags	Integer. Number of lags in VAR equations.
factors_type	Character vector. Respecting the order of the series in the input data, you must refer here the link between the (transformed) series and the factors. Three options are possible:

- "M": Variables expressed in terms of monthly growth rates can be linked to a factor representing the underlying monthly growth rate of the economy if "M" is selected
- "Q": Monthly or quarterly variables that are correlated with the the underlying quarterly growth rate of the economy can be linked to a weighted average of the factors representing the underlying monthly growth rate of the economy. Such a weighted average is meant to represent quarterly growth rates, and it is implemented by selecting "Q"
- "YoY": The variables can also be linked to the cumulative sum of the last 12 monthly factors. If the model is designed in such a way that the monthly factors represent monthly growth rates, the resulting cumulative sum boils down to the year-on-year growth rate. Thus, variables expressed in terms of year-on-year growth rates or surveys that are correlated with the year-on-year growth rates of the reference series should be linked to the factors using "YoY".

factors_loading

Boolean matrix. It represents the factor loading structure. The dimension of the matrix should be 'number of series' x 'number of factors'. For each row representing each series, the user must mention whether the corresponding factor loads on this series.

var_init

Character. The first unobserved factors values in the sample is assumed to be either equal to zero or consistent with a normal distribution with mean zero and a variance corresponding to the unconditional variance of the VAR. The latter is the default.

var_coefficients

Matrix. The default is NULL meaning that the VAR coefficients will be estimated from scratch. Alternatively, a matrix of pre-defined values can be passed in. Those would come typically from a previous model estimate and will serve as a starting point for the estimation step. The format of the matrix should be the same as the one produced by default by the create_model() function while keeping the 'var_coefficients' argument to its default value NULL.

var_errors_variance

Matrix. The default is NULL meaning that the VAR errors variance will be estimated from scratch. Alternatively, a matrix of pre-defined values can be passed in. Those would come typically from a previous model estimate and will serve as a starting point for the estimation step. The format of the matrix should be the same as the one produced by default by the create_model() function while keeping the 'var_errors_variance' argument to its default value NULL.

measurement_coefficients

Matrix. The default is NULL meaning that the measurement coefficients will be estimated from scratch. Alternatively, a matrix of pre-defined values can be passed in. Those would come typically from a previous model estimate and will serve as a starting point for the estimation step. The format of the matrix should be the same as the one produced by default by the create_model() function while keeping the 'measurement_coefficients' argument to its default value NULL.

measurement_errors_variance

Numeric vector. The default is NULL meaning that the measurement errors variance will be estimated from scratch. Alternatively, a vector of pre-defined values

can be passed in. Those would come typically from a previous model estimate and will serve as a starting point for the estimation step. The format of the vector should be the same as the one produced by default by the `create_model()` function while keeping the `'measurement_errors_variance'` argument to its default value `NULL`.

Value

an object of class `'JD3_DfmModel'`

Examples

```
# From scratch
dfm1 <- create_model(nfactors=2,
                    nlags=2,
                    factors_type = c("M", "M", "YoY", "M", "Q"),
                    factors_loading = matrix(data=TRUE, 5, 2),
                    var_init = "Unconditional")

# From a previous estimate
set.seed(100)
data<-ts(matrix(rnorm(500), 100, 5), frequency = 12, start = c(2010,1))
data[100,1]<-data[99:100,2]<-data[(1:100)[-seq(3,100,3)],5]<-NA
est1<-estimate_em(dfm1, data)

dfm2 <- create_model(nfactors=2,
                    nlags=2,
                    factors_type = c("M", "M", "YoY", "M", "Q"),
                    factors_loading = matrix(data=TRUE, 5, 2),
                    var_init = "Unconditional",
                    var_coefficients = est1$dfm$var_coefficients,
                    var_errors_variance = est1$dfm$var_errors_variance,
                    measurement_coefficients = est1$dfm$measurement_coefficients,
                    measurement_errors_variance = est1$dfm$measurement_errors_variance)
#est2<-estimate_em(dfm2, data)
```

estimate_em

Estimate DFM with Expectations-Maximization algorithm

Description

Estimate DFM with Expectations-Maximization algorithm

Usage

```
estimate_em(
  dfm,
  data,
```

```

    standardized = FALSE,
    input_standardization = NULL,
    pca_init = TRUE,
    max_iter = 100,
    eps = 1e-09,
    re_estimate = TRUE
  )

```

Arguments

dfm	an object of class 'JD3_DfmModel'. Typically generated by the create_model() function.
data	an mts object.
standardized	Boolean. Indicate whether the input series were already standardized or not. Default is FALSE, meaning that a standardization of the series will be preliminary applied as part of the process.
input_standardization	Matrix. Mean and standard deviation of the variables to consider for the pre-processing step of standardization. Default is NULL, meaning that they will be re-calculated based on the data. Typically, it can be filled with the output of the function 'get_results()\$preprocessing\$sample_mean_stdev' applied on a previous estimate of the model. If provided manually, it must be a two columns matrix with the mean in the first column and the standard deviation in the second column. In the rows, the order of the variables should also be respected (similar to the data). Note that this argument must be filled if the re_estimate argument is set to FALSE. On the other hand, it is ignored if the standardized argument is set to TRUE.
pca_init	Boolean. Indicate whether a principal components analysis is performed beforehand and used as initial condition for the EM algorithm.
max_iter	Integer. Maximum number of iterations.
eps	Numeric. EM algorithm is run until the percentage likelihood does not increase by more than the eps value (1e-9 is the default) or until the maximum number of iterations is hit.
re_estimate	Boolean. Indicate whether the model will be re-estimated or not. Default is TRUE. Could be set to FALSE if, for some reasons during the production process, we wanted to freeze to model for some periods of time. It is not recommended to freeze the model for a long period.

Value

an object of class 'JD3_DfmEstimates'

Examples

```

set.seed(100)
data<-ts(matrix(rnorm(500), 100, 5), frequency = 12, start = c(2010,1))
data[100,1]<-data[99:100,2]<-data[(1:100)[-seq(3,100,3)],5]<-NA

```

```
dfm <- create_model(nfactors=2,
                   nlags=2,
                   factors_type = c("M", "M", "YoY", "M", "Q"),
                   factors_loading = matrix(data=TRUE, 5, 2),
                   var_init = "Unconditional")
est_em<-estimate_em(dfm, data)

#est_em<-estimate_em(dfm, data, re_estimate=FALSE) # model not re-estimated
```

 estimate_ml

Estimate DFM with Maximum Likelihood

Description

Estimate DFM with Maximum Likelihood

Usage

```
estimate_ml(
  dfm,
  data,
  standardized = FALSE,
  input_standardization = NULL,
  pca_init = TRUE,
  em_init = TRUE,
  em_max_iter = 100,
  em_eps = 1e-09,
  max_iter = 1000,
  max_block_iter = 5,
  simpl_model_iter = 15,
  independent_var_shocks = FALSE,
  mixedEstimation = TRUE,
  eps = 1e-09,
  re_estimate = TRUE
)
```

Arguments

dfm	an object of class 'JD3_DfmModel'. Typically generated by the create_model() function.
data	an mts object.
standardized	Boolean. Indicate whether the input series were already standardized or not. Default is FALSE, meaning that a standardization of the series will be preliminary applied as part of the process.

input_standardization	Matrix. Mean and standard deviation of the variables to consider for the pre-processing step of standardization. Default is NULL, meaning that they will be re-calculated based on the data. Typically, it can be filled with the output of the function 'get_results()\$preprocessing\$sample_mean_stdev' applied on a previous estimate of the model. If provided manually, it must be a two columns matrix with the mean in the first column and the standard deviation in the second column. In the rows, the order of the variables should also be respected (similar to the data). Note that this argument must be filled if the re_estimate argument is set to FALSE. On the other hand, it is ignored if the standardized argument is set to TRUE.
pca_init	Boolean. Indicate whether a principal components analysis is performed beforehand and used as initial condition for either the EM algorithm (if em_init=TRUE) or directly for the ML estimation.
em_init	Boolean. Indicate whether the EM algorithm is performed beforehand and used as initial condition for the ML estimation.
em_max_iter	Integer. Maximum number of iterations of the EM algorithm. Ignored if em_init = FALSE.
em_eps	Numeric. EM algorithm is run until the percentage likelihood does not increase by more than the eps value (1e-9 is the default) or until the maximum number of iterations is hit. Ignored if em_init = FALSE.
max_iter	Integer. Maximum number of iterations for the ML estimation.
max_block_iter	Integer. Maximum number of iterations in optimization by block. The model parameters are divided in two blocks: one related to the measurement equations and one to the VAR equations. While the EM algorithm requires one iteration per block, the numerical optimization allows us to set the number of iterations desired per block.
simpl_model_iter	Integer. Number of simplified model iterations allowed.
independent_var_shocks	Boolean. Whether we assume that shocks in the VAR block are independent.
mixedEstimation	Boolean. The mixed estimation option alternates between the iterations for the VAR block alone and simultaneous iterations for the two blocks.
eps	Numeric. ML estimation is run until the percentage likelihood does not increase by more than the eps value (1e-9 is the default) or until the maximum number of iterations is hit.
re_estimate	Boolean. Indicate whether the model will be re-estimated or not. Default is TRUE. Could be set to FALSE if, for some reasons during the production process, we wanted to freeze to model for some periods of time. It is not recommended to freeze the model for a long period.

Value

an object of class 'JD3_DfmEstimates'

Examples

```
set.seed(100)
data<-ts(matrix(rnorm(500), 100, 5), frequency = 12, start = c(2010,1))
data[100,1]<-data[99:100,2]<-data[(1:100)[-seq(3,100,3)],5]<-NA
dfm <- create_model(nfactors=2,
                    nlags=2,
                    factors_type = c("M", "M", "YoY", "M", "Q"),
                    factors_loading = matrix(data=TRUE, 5, 2),
                    var_init = "Unconditional")
est_ml<-estimate_ml(dfm, data)

#est_ml<-estimate_ml(dfm, data, re_estimate=FALSE) # model not re-estimated
```

estimate_pca

Estimate DFM with Principal components Analysis

Description

Estimate DFM with Principal components Analysis

Usage

```
estimate_pca(
  dfm,
  data,
  standardized = FALSE,
  input_standardization = NULL,
  re_estimate = TRUE
)
```

Arguments

dfm	an object of class 'JD3_DfmModel'. Typically generated by the create_model() function.
data	an mts object.
standardized	Boolean. Indicate whether the input series were already standardized or not. Default is FALSE, meaning that a standardization of the series will be preliminary applied as part of the process.
input_standardization	Matrix. Mean and standard deviation of the variables to consider for the pre-processing step of standardization. Default is NULL, meaning that they will be re-calculated based on the data. Typically, it can be filled with the output of the function 'get_results()\$preprocessing\$sample_mean_stdev' applied on a previous estimate of the model. If provided manually, it must be a two columns matrix with the mean in the first column and the standard deviation in the second column. In the rows, the order of the variables should also be respected (similar

to the data). Note that this argument must be filled if the `re_estimate` argument is set to `FALSE`. On the other hand, it is ignored if the `standardized` argument is set to `TRUE`.

`re_estimate` Boolean. Indicate whether the model will be re-estimated or not. Default is `TRUE`. Could be set to `FALSE` if, for some reasons during the production process, we wanted to freeze to model for some periods of time. It is not recommended to freeze the model for a long period.

Value

an object of class 'JD3_DfmEstimates'

Examples

```
set.seed(100)
data<-ts(matrix(rnorm(500), 100, 5), frequency = 12, start = c(2010,1))
data[100,1]<-data[99:100,2]<-data[(1:100)[-seq(3,100,3)],5]<-NA
dfm <- create_model(nfactors=2,
                    nlags=2,
                    factors_type = c("M", "M", "YoY", "M", "Q"),
                    factors_loading = matrix(data=TRUE, 5, 2),
                    var_init = "Unconditional")
est_pca<-estimate_pca(dfm, data)

#est_pca<-estimate_pca(dfm, data, re_estimate=FALSE) # model not re-estimated
```

get_forecasts

Get DFM forecasts

Description

Get DFM forecasts

Usage

```
get_forecasts(dfm_estimates, n_fcst = 3)
```

Arguments

`dfm_estimates` an object of class 'JD3_DfmEstimates'

`n_fcst` Integer. Number of forecast periods required.

Value

an object of class 'JD3_DfmForecasts'

Examples

```

set.seed(100)
data<-ts(matrix(rnorm(500), 100, 5), frequency = 12, start = c(2010,1))
data[100,1]<-data[99:100,2]<-data[(1:100)[-seq(3,100,3)],5]<-NA
dfm <- create_model(nfactors=2,
                    nlags=2,
                    factors_type = c("M", "M", "YoY", "M", "Q"),
                    factors_loading = matrix(data=TRUE, 5, 2),
                    var_init = "Unconditional")
est_em<-estimate_em(dfm, data)
fcst<-get_forecasts(est_em, n_fcst = 2)

```

get_news

DFM News analysis

Description

DFM News analysis

Usage

```
get_news(dfm_estimates, new_data, target_series = NULL, n_fcst = 3)
```

Arguments

dfm_estimates an object of class 'JD3_DfmEstimates'. Typically generated by the functions estimate_pca(), estimate_em() or estimate_ml().

new_data an mts object containing the updated dataset.

target_series the name of the series of interest. By default, the first series is considered.

n_fcst the number of forecasting periods to consider. Default is 3.

Value

An object of class 'JD3_DfmNews'

References

Banbura and Modugno (2010) - Maximum likelihood estimation of factor models on data sets with arbitrary pattern of missing data

Examples

```

set.seed(100)
data_t1<-ts(matrix(rnorm(500), 100, 5), frequency = 12, start = c(2010,1))
data_t1[100,1]<-data_t1[99:100,2]<-data_t1[(1:100)[-seq(3,100,3)],5]<-NA
data_t2<-ts(rbind(data_t1, rep(NA,5)), frequency = 12, start = c(2010,1))
data_t2[100,1]<-data_t2[99,2]<-data_t2[101,3]<-data_t2[101,4]<-1

dfm_model <- create_model(nfactors=2,
                          nlags=2,
                          factors_type = c("M", "M", "YoY", "M", "Q"),
                          factors_loading = matrix(TRUE, 5, 2),
                          var_init = "Unconditional")

est_em<-estimate_em(dfm_model, data_t1)
# or to use any previous frozen model:
# est_em_frozen<-estimate_em(dfm_model, data_t1, re_estimate = FALSE)

news<-get_news(est_em, data_t2, target_series = "Series 2", n_fcst = 2)

```

get_results

Get DFM results

Description

Get DFM results

Usage

```
get_results(dfm_estimates)
```

Arguments

dfm_estimates an object of class 'JD3_DfmEstimates'

Value

an object of class 'JD3_DfmResults'

Examples

```

set.seed(100)
data<-ts(matrix(rnorm(500), 100, 5), frequency = 12, start = c(2010,1))
data[100,1]<-data[99:100,2]<-data[(1:100)[-seq(3,100,3)],5]<-NA
dfm <- create_model(nfactors=2,
                    nlags=2,
                    factors_type = c("M", "M", "YoY", "M", "Q"),
                    factors_loading = matrix(data=TRUE, 5, 2),
                    var_init = "Unconditional")

```

```
est_em<-estimate_em(dfm, data)
rslt_em<-get_results(est_em)
```

plot.JD3_DfmForecasts *Plot function for objects of class 'JD3_DfmForecasts'*

Description

Plot function for objects of class 'JD3_DfmForecasts'

Usage

```
## S3 method for class 'JD3_DfmForecasts'
plot(x, series_name = NULL, ...)
```

Arguments

x an object of class 'JD3_DfmForecasts'

series_name Character. Name of the series to plot. By default, the first series will be plotted.

... further arguments passed to ts.plot().

plot.JD3_DfmNews *Plot function for objects of class 'JD3_DfmNews'*

Description

Plot function for objects of class 'JD3_DfmNews'

Usage

```
## S3 method for class 'JD3_DfmNews'
plot(x, ...)
```

Arguments

x an object of class 'JD3_DfmNews'

... further arguments passed to barplot().

print.JD3_DfmEstimates

Print function for objects of class 'JD3_DfmEstimates'

Description

Print function for objects of class 'JD3_DfmEstimates'

Usage

```
## S3 method for class 'JD3_DfmEstimates'  
print(x, ...)
```

Arguments

x an object of class 'JD3_DfmEstimates'
... further arguments passed to the print() function.

print.JD3_DfmForecasts

Print function for objects of class 'JD3_DfmForecasts'

Description

Print function for objects of class 'JD3_DfmForecasts'

Usage

```
## S3 method for class 'JD3_DfmForecasts'  
print(x, ...)
```

Arguments

x an object of class 'JD3_DfmForecasts'
... further arguments passed to the print() function.

print.JD3_DfmNews *Print function for objects of class 'JD3_DfmNews'*

Description

Print function for objects of class 'JD3_DfmNews'

Usage

```
## S3 method for class 'JD3_DfmNews'  
print(x, ...)
```

Arguments

x an object of class 'JD3_DfmNews'
... further arguments passed to the print() function.

print.JD3_DfmResults *Print function for objects of class 'JD3_DfmResults'*

Description

Print function for objects of class 'JD3_DfmResults'

Usage

```
## S3 method for class 'JD3_DfmResults'  
print(x, ...)
```

Arguments

x an object of class 'JD3_DfmResults'
... further arguments passed to the print() function.

summary.JD3_DfmNews *Summary function for objects of class 'JD3_DfmNews'*

Description

Summary function for objects of class 'JD3_DfmNews'

Usage

```
## S3 method for class 'JD3_DfmNews'  
summary(object, ...)
```

Arguments

object an object of class 'JD3_DfmNews'
... further arguments passed to the print() function.

Index

`create_model`, [2](#)

`estimate_em`, [4](#)

`estimate_ml`, [6](#)

`estimate_pca`, [8](#)

`get_forecasts`, [9](#)

`get_news`, [10](#)

`get_results`, [11](#)

`plot.JD3_DfmForecasts`, [12](#)

`plot.JD3_DfmNews`, [12](#)

`print.JD3_DfmEstimates`, [13](#)

`print.JD3_DfmForecasts`, [13](#)

`print.JD3_DfmNews`, [14](#)

`print.JD3_DfmResults`, [14](#)

`summary.JD3_DfmNews`, [15](#)