Package: rjd3nowcasting (via r-universe)

October 23, 2026

Type Package

Title Nowcasting with Dynamic Factors Models in 'JDemetra+' 3.x

Version 2.0.3.9000

Description Interface to 'JDemetra+' 3.x (<https://github.com/jdemetra>), time series analysis software. The package enables the specification and estimation of Dynamic Factor Models for nowcasting, and includes news analysis to interpret forecast revisions. It implements highly efficient algorithms for fast and reliable computation.

License EUPL

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

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

Depends R (>= 4.1.0)

Imports rJava (>= 1.0-6), rjd3jars (>= 0.0.5), rjd3toolkit (>= 3.8.0), stats, graphics, grDevices

Encoding UTF-8

Roxygen list(markdown = TRUE)

SystemRequirements Java (>= 21)

Collate 'utils.R' 'estimation.R' 'model.R' 'news.R' 'print.R' 'results.R' 'zzz.R'

LazyData TRUE

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.3.3

Config/pak/sysreqs make default-jdk libprotobuf-dev protobuf-compiler libprotoc-dev

Repository https://tanguybarthelemy.r-universe.dev

Date/Publication 2026-08-12 09:38:06 UTC

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

RemoteRef HEAD

RemoteSha 74c643738e320f7549125613eeca8bb9caebc53b

Contents

create_model 2
estimate_em 4
estimate_ml 7
estimate_pca 9
get_forecasts 11
get_news 13
get_results 15
macroIndicators 16

Index **17**

`create_model` *Create a Dynamic Factor Model*

Description

Creates a Dynamic Factor Model (DFM) by specifying the number of factors and the number of lags in the Vector Auto-Regressive (VAR) process. The function also allows the user to choose between different types of links between each series and the latent factors, in addition to defining the factor loading structure.

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 the VAR process.
`factors_type`	Character vector. Defines, in the order of the input series, how each (transformed) series relates to the factors. Available options are:

- "M": Variables expressed as monthly growth rates, linked directly to the latent factors.
- "Q": Quarterly or monthly variables linked to a weighted average of the factors, representing quarterly growth rates.
- "YoY": Variables linked to the cumulative sum of the last 12 monthly factors, corresponding to year-on-year growth rates. This option is appropriate for variables expressed in year-on-year terms, or for series that are closely related to such evolution.

`factors_loading`
: Boolean matrix defining the factor loading structure, with dimension *number of series* × *number of factors*. Each entry indicates whether a given factor loads on a given series.

`var_init`
: Character. Specifies the initialization of the latent factors: either zero or assumed to follow a normal distribution with mean zero and variance equal to the unconditional variance of the VAR (default).

`var_coefficients`
: Matrix. The VAR coefficients. If `NULL` (default), they will be estimated from scratch. Otherwise, user-defined values can be provided as starting point for the estimation, typically obtained from a previous estimation of the model.

`var_errors_variance`
: Matrix. The variance-covariance matrix of the VAR errors. If `NULL` (default), it will be estimated from scratch. Otherwise, user-defined values can be provided as starting point for the estimation, typically obtained from a previous estimation of the model.

`measurement_coefficients`
: Matrix. The measurement equation coefficients. If `NULL` (default), they will be estimated from scratch. Otherwise, user-defined values can be provided as starting point for the estimation, typically obtained from a previous estimation of the model.

`measurement_errors_variance`
: Numeric vector. The variance of the idiosyncratic measurement errors. If `NULL` (default), they will be estimated from scratch. Otherwise, user-defined values can be provided as starting point for the estimation, typically obtained from a previous estimation of the model.

Value

An object of class `"JD3_DFMMODEL"` is returned. The following are returned invisibly as a list:

- `var_coefficients` `[[1]]` initial values of the VAR coefficients;
- `var_errors_variance` `[[2]]` initial values of the variance-covariance matrix of the VAR errors;
- `measurement_coefficients` `[[3]]` initial values of the measurement equation coefficients;
- `measurement_errors_variance` `[[4]]` initial values of the variance of the idiosyncratic measurement errors;
- `initialization_type` `[[5]]` the value of the `var_init` argument;
- `factors_type` `[[6]]` the value of the `factors_type` argument.

See Also

`estimate_pca()` to estimate the generated model using principal components analysis,

`estimate_em()` to estimate the generated model using the Expectations-Maximization algorithm,

`estimate_ml()` to estimate the generated model using maximum likelihood.

For more information, see the vignette:

`utils::browseVignettes()`, e.g. `browseVignettes(package = "rjd3nowcasting")`

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 previously estimated model
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)
```

---

estimate_em — *Estimate a Dynamic Factor Model Using the Expectations-Maximization Algorithm*

Description

Estimate the model parameters using the Expectations-Maximization (EM) algorithm, with PCA-based initialization by default. The function includes optional arguments to tune the estimation process.

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 using the `create_model()` function.

`data` A `"mts"` object containing the (transformed) input data.

`standardized` Boolean. Indicates whether the input data are already standardized. The default is `FALSE`, in which case standardization is applied as a preprocessing step.

`input_standardization` A matrix specifying the mean and standard deviation of the series to be used during standardization. The default is `NULL`, meaning that these quantities are computed from the data. This argument can be set using the output of `get_results()$preprocessing$sampl` from a previous model estimation. If provided manually, it must be a two-column matrix with the means in the first column and the standard deviations in the second column. The ordering of the rows must match the series in `data`. This argument must be provided if `re_estimate = FALSE`, and is ignored if `standardized = TRUE`.

`pca_init` Boolean. Indicates whether a Principal Components Analysis (PCA) is performed beforehand and used to initialize the EM algorithm. The default is `TRUE`.

`max_iter` Integer. Specifies the maximum number of iterations in the EM algorithm. The default is `100`.

`eps` Numeric. The EM algorithm runs until the increase of the percentage likelihood falls below the `eps` value (default is `1e-9`), or when the maximum number of iterations is reached.

`re_estimate` Boolean. Indicates whether the model parameters should be re-estimated. The default is `TRUE`. It can be set to `FALSE` to keep the model frozen for some period of time, although prolonged use of a frozen model is not recommended.

Value

An object of class `"JD3_DFMESTIMATES"` is returned. The following are returned invisibly as a list:

- `dfm [[1]]` an object of class `"JD3_DFMMODEL"` containing the estimated model parameters;
- `data [[2]]` the value of the `data` argument;
- `is_standardized [[3]]` the value of the `standardized` argument;

- `input_standardization` `[[4]]` the value of the `input_standardization` argument;
- `log_likelihood` `[[5]]` the estimated log-likelihood;
- a couple of other elements not relevant for the EM estimates;
- `has_converged` `[[8]]` a boolean indicating whether the EM algorithm has converged.

See Also

`create_model()` to define a new model,

`estimate_pca()` for estimation using principal components analysis,

`estimate_ml()` for estimation using maximum likelihood,

`get_results()` to access estimation results,

`get_forecasts()` to obtain forecasts.

For more information, see the vignette:

`utils::browseVignettes()`, e.g. `browseVignettes(package = "rjd3nowcasting")`

Examples

```
# input data
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

# define a new model
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"
)

# estimate using EM algorithm
est_em_a <- estimate_em(dfm, data)

# results and forecasts
rslts_a <- get_results(est_em_a)
fcsts_a <- get_forecasts(est_em_a)

# no re-estimation of a previous model while integrating updated data in the output object
data_new <- data
data_new[99, 2] <- 1
est_em_b <- estimate_em(est_em_a$dfm,
                        data_new,
                        input_standardization = rslts_a$preprocessing$sample_mean_stdev,
                        re_estimate = FALSE)
```

estimate_ml *Estimate a Dynamic Factor Model by Maximum Likelihood*

Description

Estimate the model parameters by Maximum Likelihood (ML), with initial values obtained from the Expectations-Maximization (EM) algorithm by default. The function includes optional arguments to tune the estimation process.

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 using the `create_model()` function.

`data` A `"mts"` object containing the (transformed) input data.

`standardized` Boolean. Indicates whether the input data are already standardized. The default is `FALSE`, in which case standardization is applied as a preprocessing step.

`input_standardization` A matrix specifying the mean and standard deviation of the series to be used during standardization. The default is `NULL`, meaning that these quantities are computed from the data. This argument can be set using the output of `get_results()$preprocessing$sampl` from a previous model estimation. If provided manually, it must be a two-column matrix with the means in the first column and the standard deviations in the second column. The ordering of the rows must match the series in `data`. This argument must be provided if `re_estimate = FALSE`, and is ignored if `standardized = TRUE`.

`pca_init` Boolean. Indicates whether a Principal Components Analysis (PCA) is performed beforehand and used to initialize either the EM algorithm (if `em_init = TRUE`) or directly the ML estimation. The default is `TRUE`.

`em_init` Boolean. Indicates whether the EM algorithm is run prior to ML estimation and used to provide initial values. The default is `TRUE`.

`em_max_iter` Integer. Specifies the maximum number of iterations for the EM algorithm. Ignored if `em_init = FALSE`. The default is `100`.

`em_eps` Numeric. The EM algorithm runs until the increase of the percentage likelihood falls below the `eps` value (default is `1e-9`), or when the maximum number of iterations is reached. Ignored if `em_init = FALSE`.

`max_iter` Integer. Specifies the maximum number of iterations for the ML estimation. The default is `1000`.

`max_block_iter` Integer. Specifies the maximum number of iterations per block in the optimization process. Model parameters are divided into two blocks: one for measurement equation and one for VAR equations. Unlike the EM algorithm (one iteration per block), numerical optimization allows multiple iterations per block. The default is `5`.

`simpl_model_iter` Integer. Specifies the number of iterations allowed for the simplified model. The default is `15`.

`independent_var_shocks` Boolean. Indicates whether shocks in the VAR block are assumed to be independent. The default is `FALSE`.

`mixedEstimation` Boolean. Indicates whether to alternate between iterations on the VAR block alone and simultaneous iterations on the two blocks. The default is `TRUE`.

`eps` Numeric. The ML estimation runs until the increase of the percentage likelihood falls below the `eps` value (default is `1e-9`), or when the maximum number of iterations is reached.

`re_estimate` Boolean. Indicates whether the model parameters should be re-estimated. The default is `TRUE`. It can be set to `FALSE` to keep the model frozen for some period of time, although prolonged use of a frozen model is not recommended.

Value

An object of class `"JD3_DFMESTIMATES"` is returned. The following are returned invisibly as a list:

- `dfm` `[[1]]` an object of class `"JD3_DFMMODEL"` containing the estimated model parameters;
- `data` `[[2]]` the value of the `data` argument;
- `is_standardized` `[[3]]` the value of the `standardized` argument;
- `input_standardization` `[[4]]` the value of the `input_standardization` argument;
- `log_likelihood` `[[5]]` the estimated log-likelihood;
- `gradient` `[[6]]` the estimated gradient at the solution;
- `hessian` `[[7]]` the estimated hessian matrix at the solution;
- `has_converged` `[[8]]` a boolean indicating whether the ML estimation process has converged.

See Also

`create_model()` to define a new model,

`estimate_pca()` for estimation using principal components analysis,

`estimate_em()` for estimation using EM algorithm,

`get_results()` to access estimation results,

`get_forecasts()` to obtain forecasts.

For more information, see the vignette:

`utils::browseVignettes()`, e.g. `browseVignettes(package = "rjd3nowcasting")`

Examples

```
# input data
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

# define a new model
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"
)

# estimate by maximum likelihood
est_ml_a <- estimate_ml(dfm, data)

# results and forecasts
rslts_a <- get_results(est_ml_a)
fcsts_a <- get_forecasts(est_ml_a)

# no re-estimation of a previous model while integrating updated data in the output object
data_new <- data
data_new[99, 2] <- 1
est_ml_b <- estimate_ml(est_ml_a$dfm,
                        data_new,
                        input_standardization = rslts_a$preprocessing$sample_mean_stdev,
                        re_estimate = FALSE)
```

`estimate_pca` *Estimate a Dynamic Factor Model Using Principal Components Analysis*

Description

Estimate the model parameters using Principal Component Analysis (PCA). While this approach is fast, it is generally not recommended to rely on it exclusively, particularly when the dataset includes quarterly series or variables related to year-on-year growth rates.

Usage

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

Arguments

`dfm` An object of class `"JD3_DFMMODEL"`, typically generated using the `create_model()` function.

`data` A `"mts"` object containing the (transformed) input data.

`standardized` Boolean. Indicates whether the input data are already standardized. The default is `FALSE`, in which case standardization is applied as a preprocessing step.

`input_standardization` A matrix specifying the mean and standard deviation of the series to be used during standardization. The default is `NULL`, meaning that these quantities are computed from the data. This argument can be set using the output of `get_results()$preprocessing$sampl` from a previous model estimation. If provided manually, it must be a two-column matrix with the means in the first column and the standard deviations in the second column. The ordering of the rows must match the series in `data`. This argument must be provided if `re_estimate = FALSE`, and is ignored if `standardized = TRUE`.

`re_estimate` Boolean. Indicates whether the model parameters should be re-estimated. The default is `TRUE`. It can be set to `FALSE` to keep the model frozen for some period of time, although prolonged use of a frozen model is not recommended.

Value

An object of class `"JD3_DFMESTIMATES"` is returned. The following are returned invisibly as a list:

- `dfm [[1]]` an object of class `"JD3_DFMMODEL"` containing the estimated model parameters;
- `data [[2]]` the value of the `data` argument;
- `is_standardized [[3]]` the value of the `standardized` argument;
- `input_standardization [[4]]` the value of the `input_standardization` argument;
- additional elements not relevant for Principal Components Analysis estimates.

See Also

`create_model()` to define a new model,

`estimate_em()` for estimation using the Expectations-Maximization algorithm,

`estimate_ml()` for estimation using maximum likelihood,

`get_results()` to access estimation results,

`get_forecasts()` to obtain forecasts.

For more information, see the vignette:

`utils::browseVignettes()`, e.g. `browseVignettes(package = "rjd3nowcasting")`

Examples

```
# input data
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

# define a new model
dfm <- create_model(
    nfactors = 2,
    nlags = 2,
    factors_type = c("M", "M", "M", "M", "M"),
    factors_loading = matrix(data = TRUE, 5, 2),
    var_init = "Unconditional"
)

# estimate using PCA
est_pca_a <- estimate_pca(dfm, data)

# results and forecasts
rslts_a <- get_results(est_pca_a)
fcsts_a <- get_forecasts(est_pca_a)

# no re-estimation of a previous model while integrating updated data in the output object
data_new <- data
data_new[99, 2] <- 1
est_pca_b <- estimate_pca(est_pca_a$dfm,
                          data_new,
                        input_standardization = rslts_a$preprocessing$sample_mean_stdev,
                          re_estimate = FALSE)
```

`get_forecasts` *Get Dynamic Factor Model Forecasts*

Description

Provides access to forecasts and their associated standard deviations for both the original and transformed series. Optional argument allows to include the missing values estimate in the output.

Usage

```
get_forecasts(
  dfm_estimates,
  n_fcst = 3,
  estim_missing = FALSE,
  mask_q_m = FALSE
)
```

Arguments

`dfm_estimates`	An object of class `"JD3_DFMESTIMATES"`, typically generated using the `estimate_ml()`, `estimate_em()`, or `estimate_pca()` function.
`n_fcst`	Integer. Number of forecast periods to consider. The default is `3`.
`estim_missing`	Boolean. Indicates whether missing values should be estimated prior to the start of the forecasting period. The default is `FALSE`.
`mask_q_m`	Boolean. Indicates whether estimates for the first two months of each quarter should be masked when the factor type is `Q`. The default is `FALSE`.

Value

An object of class `"JD3_DFMFORECASTS"` is returned. The following are returned invisibly as a list:

- `transformed_forecasts` `[[1]]` the transformed series together with their forecasts;
- `transformed_forecasts_stdev` `[[2]]` standard deviations of the transformed series and their forecasts;
- `forecasts` `[[3]]` the original series together with their forecasts;
- `forecasts_stdev` `[[4]]` standard deviations of the original series and their forecasts;
- `forecasts_only` `[[5]]` the forecasts of the original series;
- `forecasts_only_stdev` `[[6]]` standard deviations of the forecasts of the original series.

See Also

`get_results()` to obtain estimation results.

For more information, see the vignette:

`utils::browseVignettes()`, e.g. `browseVignettes(package = "rjd3nowcasting")`

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)

fcsts_em <- get_forecasts(est_em, n_fcst = 2)
```

`get_news` *Perform News Analysis for Forecast Updates in Dynamic Factor Models*

Description

Performs a news analysis to quantify the impact of newly released data on forecast revisions between two consecutive dataset updates. The function isolates the contribution of new observations by comparing them to their forecasts obtained from the revised dataset (including past data revisions) and provides a detailed decomposition of the resulting forecast changes.

Usage

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

Arguments

`dfm_estimates`	An object of class `"JD3_DFMESTIMATES"`, typically generated using the `estimate_ml()`, `estimate_em()`, or `estimate_pca()` function.
`new_data`	A `"mts"` object containing the updated dataset.
`target_series`	Character. Name of the target series of interest. By default, the first series is used.
`n_fcst`	Integer. Number of forecast periods to consider. The default is `3`.

Value

An object of class "JD3_DFMNEWS" is returned. The following are returned invisibly as a list:

- target_series [[1]] the value of the target_series argument;
- weights_T [[2]] the weights of the news for the transformed series;
- impacts_T [[3]] the impacts of the news (weights x news) for the transformed series;
- forecasts_T [[4]] the old, revised (including past data revisions) and new forecasts of the transformed series;
- weights [[5]] the weights of the news for the original series;
- impacts [[6]] the impacts of the news (weights x news) for the original series;
- forecasts [[7]] the old, revised (including past data revisions) and new forecasts of the original series.

References

Banbura, M., & Modugno, M. (2010). Maximum likelihood estimation of factor models on data sets with arbitrary patterns of missing data.

See Also

create_model() to define a new Dynamic Factor model,

estimate_pca() for estimation using principal components analysis,

estimate_em() for estimation using EM algorithm,

estimate_ml() for estimation using maximum likelihood,

For more information, see the vignette:

utils::browseVignettes(), e.g. browseVignettes(package = "rjd3nowcasting")

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_a <- estimate_em(dfm_model, data_t1)

news_a <- get_news(est_em_a, data_t2, target_series = "Series 2", n_fcst = 2)
```

`get_results` *Get Dynamic Factor Model Results*

Description

Provides access to estimation results, including pre-processing (e.g., standardization input for dynamic workflows), parameter and factor estimates, residuals, and likelihood.

Usage

```
get_results(dfm_estimates)
```

Arguments

`dfm_estimates`	An object of class `"JD3_DFMESTIMATES"`, typically generated using the `estimate_ml()`, `estimate_em()`, or `estimate_pca()` function.

Value

An object of class `"JD3_DFMRESULTS"` is returned. The following are returned invisibly as a list:

- `preprocessing` `[[1]]` the original and transformed data, along with the sample mean and standard deviation used for standardization;
- `parameters` `[[2]]` the estimated parameters, the variance of the measurement errors, and the variance-covariance matrix of the VAR errors;
- `factors` `[[3]]` the estimated factors and their standard deviations;
- `residuals` `[[4]]` the residuals and the standardized residuals for diagnostic checks;
- `likelihood` `[[5]]` the estimated log-likelihood and a Boolean indicating whether the estimation has converged.

See Also

`get_forecasts()` to obtain forecast results.

For more information, see the vignette:

`utils::browseVignettes()`, e.g. `browseVignettes(package = "rjd3nowcasting")`

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)
```

`macroIndicators` *Datasets including some French macro-economic variables*

Description

The datasets 'data0' and 'data1' acts as successive releases of macro-economic time series. They contain data on monthly industrial production index (PVI), turnover (TURN), quarterly GDP, as well as business survey data (BS) and other survey data (PMI) for both France and the Eurozone. Those datasets are used to illustrate how one of these variable can be nowcasted using the others using a Dynamic Factor model.

Usage

```
data0

data1
```

Format

An object of class `data.frame` with 150 rows and 11 columns.

An object of class `data.frame` with 150 rows and 11 columns.

Index

∗ **datasets**
    macroIndicators, 16

create_model, 2

data0 *(macroIndicators)*, 16
data1 *(macroIndicators)*, 16

estimate_em, 4
estimate_ml, 7
estimate_pca, 9

get_forecasts, 11
get_news, 13
get_results, 15

macroIndicators, 16