Package 'dfms'

May 18, 2025

Version 0.3.0

Title Dynamic Factor Models

Description Efficient estimation of Dynamic Factor Models using the Expectation Maximization (EM) algorithm
or Two-Step (2S) estimation, supporting datasets with missing data. Factors are assumed to follow a stationary VAR
process of order p. The estimation options follow advances in the econometric literature: either running the Kalman
Filter and Smoother once with initial values from PCA -
2S estimation as in Doz, Giannone and Reichlin (2011)
<doi:10.1016/j.jeconom.2011.02.012> -
or via iterated Kalman Filtering and Smoothing until EM convergence - following
Doz, Giannone and Reichlin (2012) <doi:10.1162/REST_a_00225> -
or using the adapted EM algorithm of Banbura and
Modugno (2014) <doi:10.1002/jae.2306>, allowing arbitrary patterns of missing data. The implementation makes heavy
use of the 'Armadillo' 'C++' library and the 'collapse' package, providing for particularly speedy estimation.
A comprehensive set of methods supports interpretation and visualization of the model as well as forecasting.
Information criteria to choose the number of factors are also provided -
following Bai and Ng (2002)
<doi:10.1111/1468-0262.00273>.

URL https://sebkrantz.github.io/dfms/

BugReports https://github.com/SebKrantz/dfms/issues

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0.1), collapse (>= 2.0.0)

LinkingTo Rcpp, RcppArmadillo

Suggests xts, vars, magrittr, testthat (>= 3.0.0), knitr, rmarkdown, covr

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation yes

Author Sebastian Krantz [aut, cre],
      Rytis Bagdziunas [aut],
      Santtu Tikka [rev],
      Eli Holmes [rev]

Maintainer Sebastian Krantz <sebastian.krantz@graduateinstitute.ch>

Repository CRAN

Date/Publication 2025-05-18 04:50:02 UTC

Contents

dfms-package	*Dynamic Factor Models*

Description

dfms provides efficient estimation of Dynamic Factor Models via the EM Algorithm — following
Doz, Giannone & Reichlin (2011, 2012) and Banbura & Modugno (2014). Contents:

Information Criteria to Determine the Number of Factors

ICr()

- plot(<ICr>)

- screeplot(<ICr>)

Fit a Dynamic Factor Model

DFM()

- summary(<dfm>)

- plot(<dfm>)

- as.data.frame(<dfm>)

- residuals(<dfm>)

- fitted(<dfm>)

Generate Forecasts

predict(<dfm>)

- plot(<dfm_forecast>)

- as.data.frame(<dfm_forecast>)

Fast Stationary Kalman Filtering and Smoothing

SKF() — Stationary Kalman Filter
FIS() — Fixed Interval Smoother
SKFS() — Stationary Kalman Filter + Smoother

Helper Functions

.VAR() — (Fast) Barebones Vector-Autoregression
ainv() — Armadillo's Inverse Function
apinv() — Armadillo's Pseudo-Inverse Function
tsnarmimp() — Remove and Impute Missing Values in a Multivariate Time Series
em_converged() — Convergence Test for EM-Algorithm

Data

BM14_M — Monthly Series by Banbura and Modugno (2014)
BM14_Q — Quarterly Series by Banbura and Modugno (2014)
BM14_Models — Series Metadata + Small/Medium/Large Model Specifications

References

Doz, C., Giannone, D., & Reichlin, L. (2011). A two-step estimator for large approximate dynamic factor models based on Kalman filtering. *Journal of Econometrics, 164*(1), 188-205. doi:10.1016/j.jeconom.2011.02.012

Doz, C., Giannone, D., & Reichlin, L. (2012). A quasi-maximum likelihood approach for large, approximate dynamic factor models. *Review of Economics and Statistics, 94*(4), 1014-1024. doi:10.1162/REST_a_00225

Banbura, M., & Modugno, M. (2014). Maximum likelihood estimation of factor models on datasets with arbitrary pattern of missing data. *Journal of Applied Econometrics, 29*(1), 133-160. doi:10.1002/jae.2306

.VAR   *(Fast) Barebones Vector-Autoregression*

Description

Quickly estimate a VAR(p) model using Armadillo's inverse function.

Usage

```
.VAR(x, p = 1L)
```

Arguments

x	data numeric matrix with time series in columns - without missing values.
p	positive integer. The lag order of the VAR.

Value

A list containing matrices Y = x[-(1:p),], X which contains lags 1 - p of x combined column-wise, A which is the $np \times n$ transition matrix, where n is the number of series in x, and the VAR residual matrix res = Y - X %*% A.

A list with the following elements:

Y	x[-(1:p),].
X	lags 1 - p of x combined column-wise.
A	$np \times n$ transition matrix, where n is the number of series in x.
res	VAR residual matrix: Y - X %*% A.

See Also

dfms-package

Examples

```
var = .VAR(diff(EuStockMarkets), 3)
str(var)
var$A
rm(var)
```

---

ainv   *Armadillo's Inverse Functions*

Description

Matrix inverse and pseudo-inverse by the Armadillo C++ library.

Usage

```
ainv(x)

apinv(x)
```

Arguments

x   a numeric matrix, must be square for `ainv`.

Value

The matrix-inverse or pseudo-inverse.

See Also

[dfms-package](#)

Examples

```
ainv(crossprod(diff(EuStockMarkets)))
```

as.data.frame.dfm   *Extract Factor Estimates in a Data Frame*

Description

Extract Factor Estimates in a Data Frame

Usage

```
## S3 method for class 'dfm'
as.data.frame(
  x,
  ...,
  method = "all",
  pivot = c("long", "wide.factor", "wide.method", "wide", "t.wide"),
  time = seq_row(x$F_pca),
  stringsAsFactors = TRUE
)
```

Arguments

x	an object class 'dfm'.
...	not used.
method	character. The factor estimates to use: any of "qml", "2s", "pca" (multiple can be supplied) or "all" for all estimates.
pivot	character. The orientation of the frame: "long", "wide.factor" or "wide.method", "wide" or "t.wide".
time	a vector identifying the time dimension, or NULL to omit a time variable.
stringsAsFactors	
	make factors from method and factor identifiers. Same as option to as.data.frame.table.

Value

A data frame of factor estimates.

See Also

dfms-package

Examples

```
library(xts)
# Fit DFM with 3 factors and 3 lags in the transition equation
mod <- DFM(diff(BM14_M), r = 3, p = 3)

# Taking a single estimate:
print(head(as.data.frame(mod, method = "qml")))
```

```
print(head(as.data.frame(mod, method = "qml", pivot = "wide")))

# Adding a proper time variable
time <- index(BM14_M)[-1L]
print(head(as.data.frame(mod, method = "qml", time = time)))

# All estimates: different pivoting methods
for (pv in c("long", "wide.factor", "wide.method", "wide", "t.wide")) {
   cat("\npivot = ", pv, "\n")
   print(head(as.data.frame(mod, pivot = pv, time = time), 3))
}
```

BM14_Models   *Euro Area Macroeconomic Data from Banbura and Modugno 2014*

Description

A data extract from BM 2014 replication files. Some proprietary series (mostly PMI's) are excluded. The dataset BM14_Models provides information about all series and their inclusion in the 'small', 'medium' and 'large' sized dynamic factor models estimated by BM 2014. The actual data is contained in *xts* format in BM14_M for monthly data and BM14_Q for quarterly data.

Usage

```
BM14_Models
BM14_M
BM14_Q
```

Format

BM14_Models is a data frame with 101 obs. (series) and 8 columns:

series BM14 series code (converted to snake case for R)

label BM14 series label

code original series code from data source

freq series frequency

log_trans logical indicating whether the series was transformed by the natural log before differencing. Note that all data are provided in untransformed levels, and all data was (log-)differenced by BM14 before estimation.

small logical indicating series included in the 'small' model of BM14. Proprietary series are excluded.

medium logical indicating series included in the 'medium' model of BM14. Proprietary series are excluded.

large logical indicating series included in the 'large' model of BM14. This comprises all series, thus the variable is redundant but included for completeness. Proprietary series are excluded.

Source

Banbura, M., & Modugno, M. (2014). Maximum likelihood estimation of factor models on datasets with arbitrary pattern of missing data. *Journal of Applied Econometrics, 29*(1), 133-160.

See Also

[dfms-package](#)

Examples

```
library(magrittr)
library(xts)

# Constructing the database for the large model
BM14 = merge(BM14_M, BM14_Q)
BM14[, BM14_Models$log_trans] %<>% log()
BM14[, BM14_Models$freq == "M"] %<>% diff()
BM14[, BM14_Models$freq == "Q"] %<>% diff(3)

# Small Model Database
head(BM14[, BM14_Models$small])

# Medium-Sized Model Database
head(BM14[, BM14_Models$medium])
```

DFM   *Estimate a Dynamic Factor Model*

Description

Efficient estimation of a Dynamic Factor Model via the EM Algorithm - on stationary data with time-invariant system matrices and classical assumptions, while permitting missing data.

Usage

```
DFM(
  X,
  r,
  p = 1L,
  ...,
  idio.ar1 = FALSE,
  quarterly.vars = NULL,
  rQ = c("none", "diagonal", "identity"),
  rR = c("diagonal", "identity", "none"),
  em.method = c("auto", "DGR", "BM", "none"),
  min.iter = 25L,
  max.iter = 100L,
```

```
    tol = 1e-04,
    pos.corr = TRUE,
    check.increased = FALSE
)
```

Arguments

X	a T x n numeric data matrix or frame of stationary time series. May contain missing values. *Note* that data is internally standardized (scaled and centered) before estimation.
r	integer. Number of factors.
p	integer. Number of lags in factor VAR.
...	(optional) arguments to `tsnarmimp`. The default settings impute internal missing values with a cubic spline and the edges with the median and a 3-period moving average.
idio.ar1	logical. Model observation errors as AR(1) processes: $e_t = \rho e_{t-1} + v_t$. *Note* that this substantially increases computation time, and is generally not needed if n is large (>30). See theoretical vignette for details.
quarterly.vars	character. Names of quarterly variables in X (if any). Monthly variables should be to the left of the quarterly variables in the data matrix and quarterly observations should be provided every 3rd period.
rQ	character. Restrictions on the state (transition) covariance matrix (Q).
rR	character. Restrictions on the observation (measurement) covariance matrix (R).
em.method	character. The implementation of the Expectation Maximization Algorithm used. The options are:

"auto"	Automatic selection: "BM" if anyNA(X), else "DGR".
"DGR"	The classical EM implementation of Doz, Giannone and Reichlin (2012). This implementation is efficient and qu
"BM"	The modified EM algorithm of Banbura and Modugno (2014) which also accounts for missing data in the EM ite
"none"	Performs no EM iterations and just returns the Two-Step estimates from running the data through the Kalman Fil

min.iter	integer. Minimum number of EM iterations (to ensure a convergence path).
max.iter	integer. Maximum number of EM iterations.
tol	numeric. EM convergence tolerance.
pos.corr	logical. Increase the likelihood that factors correlate positively with the data, by scaling the eigenvectors such that the principal components (used to initialize the Kalman Filter) co-vary positively with the row-means of the standardized data.
check.increased	
	logical. Check if likelihood has increased. Passed to `em_converged`. If TRUE, the algorithm only terminates if convergence was reached with decreasing likelihood.

Details

This function efficiently estimates a Dynamic Factor Model with the following classical assumptions:

1. Linearity

2. Idiosynchratic measurement (observation) errors (*R* is diagonal)

3. No direct relationship between series and lagged factors (*ceteris paribus* contemporaneous factors)

4. No relationship between lagged error terms in the either measurement or transition equation (no serial correlation), unless explicitly modeled as AR(1) processes using `idio.ar1 = TRUE`.

Factors are allowed to evolve in a $VAR(p)$ process, and data is internally standardized (scaled and centered) before estimation (removing the need of intercept terms). By assumptions 1-4, this translates into the following dynamic form:

$$\mathbf{x}_t = \mathbf{C}_0 \mathbf{f}_t + \mathbf{e}_t \ \sim \ N(\mathbf{0}, \mathbf{R})$$

$$\mathbf{f}_t = \sum_{j=1}^{p} \mathbf{A}_j \mathbf{f}_{t-j} + \mathbf{u}_t \ \sim \ N(\mathbf{0}, \mathbf{Q}_0)$$

where the first equation is called the measurement or observation equation and the second equation is called transition, state or process equation, and

n number of series in $\mathbf{x}_t$ (*r* and *p* as the arguments to DFM).

$\mathbf{x}_t$ $n \times 1$ vector of observed series at time t: $(x_{1t}, \dots, x_{nt})'$. Some observations can be missing.

$\mathbf{f}_t$ $r \times 1$ vector of factors at time t: $(f_{1t}, \dots, f_{rt})'$.

$\mathbf{C}_0$ $n \times r$ measurement (observation) matrix.

$\mathbf{A}_j$ $r \times r$ state transition matrix at lag j.

$\mathbf{Q}_0$ $r \times r$ state covariance matrix.

$\mathbf{R}$ $n \times n$ measurement (observation) covariance matrix. It is diagonal by assumption 2 that $E[\mathbf{x}_{it}|\mathbf{x}_{-i,t}, \mathbf{x}_{i,t-1}, \dots, \mathbf{f}_t, \mathbf{f}_t$

This model can be estimated using a classical form of the Kalman Filter and the Expectation Maximization (EM) algorithm, after transforming it to State-Space (stacked, VAR(1)) form:

$$\mathbf{x}_t = \mathbf{C}\mathbf{F}_t + \mathbf{e}_t \ \sim \ N(\mathbf{0}, \mathbf{R})$$

$$\mathbf{F}_t = \mathbf{A}\,\mathbf{F}_{t-1} + \mathbf{u}_t \ \sim \ N(\mathbf{0}, \mathbf{Q})$$

where

n	number of series in $\mathbf{x}_t$ (r and p as the arguments to DFM).
$\mathbf{x}_t$	$n \times 1$ vector of observed series at time t: $(x_{1t}, \ldots, x_{nt})'$. Some observations can be missing.
$\mathbf{F}_t$	$rp \times 1$ vector of stacked factors at time t: $(f_{1t}, \ldots, f_{rt}, f_{1,t-1}, \ldots, f_{r,t-1}, \ldots, f_{1,t-p}, \ldots, f_{r,t-p})'$.
$\mathbf{C}$	$n \times rp$ observation matrix. Only the first $n \times r$ terms are non-zero, by assumption 3 that $E[\mathbf{x}_t|\mathbf{F}_t] = E[\mathbf{x}_t|\mathbf{f}_t]$ (no relat
$\mathbf{A}$	stacked $rp \times rp$ state transition matrix consisting of 3 parts: the top $r \times rp$ part provides the dynamic relationships ca
$\mathbf{Q}$	$rp \times rp$ state covariance matrix. The top $r \times r$ part gives the contemporaneous relationships, the rest are zeros by assu
$\mathbf{R}$	$n \times n$ observation covariance matrix. It is diagonal by assumption 2 and identical to $\mathbf{R}$ as stated in the dynamic form.

The filter is initialized with PCA estimates on the imputed dataset—see SKFS for a complete code example.

Value

A list-like object of class 'dfm' with the following elements:

X_imp	$T \times n$ matrix with the imputed and standardized (scaled and centered) data— after applying tsnarmimp. It has attributes attached allowing for reconstruction of the original data:
"stats"	is a $n \times 5$ matrix of summary statistics of class "qsu" (see qsu).
"missing"	is a $T \times n$ logical matrix indicating missing or infinite values in the original data (which are imputed in X
"attributes"	contains the attributes of the original data input.
"is.list"	is a logical value indicating whether the original data input was a list / data frame.

eigen	eigen(cov(X_imp)).
F_pca	$T \times r$ matrix of principal component factor estimates - X_imp %*% eigen$vectors.
P_0	$r \times r$ initial factor covariance matrix estimate based on PCA results.
F_2s	$T \times r$ matrix two-step factor estimates as in Doz, Giannone and Reichlin (2011) - obtained from running the data through the Kalman Filter and Smoother once, where the Filter is initialized with results from PCA.
P_2s	$r \times r \times T$ covariance matrices of two-step factor estimates.
F_qml	$T \times r$ matrix of quasi-maximum likelihood factor estimates - obtained by iteratively Kalman Filtering and Smoothing the factor estimates until EM convergence.

P_qml	$r \times r \times T$ covariance matrices of QML factor estimates.
A	$r \times rp$ factor transition matrix.
C	$n \times r$ observation matrix.
Q	$r \times r$ state (error) covariance matrix.
R	$n \times n$ observation (error) covariance matrix.
e	$T \times n$ estimates of observation errors $\mathbf{e}_t$. Only available if idio.ar1 = TRUE.
rho	$n \times 1$ estimates of AR(1) coefficients (ρ) in observation errors: $e_t = \rho e_{t-1} + v_t$. Only available if idio.ar1 = TRUE.
loglik	vector of log-likelihoods - one for each EM iteration. The final value corresponds to the log-likelihood of the reported model.
tol	The numeric convergence tolerance used.
converged	single logical valued indicating whether the EM algorithm converged (within max.iter iterations subject to tol).
anyNA	single logical valued indicating whether there were any (internal) missing values in the data (determined after removal of rows with too many missing values). If FALSE, X_imp is simply the original data in matrix form, and does not have the "missing" attribute attached.
rm.rows	vector of any cases (rows) that were removed beforehand (subject to max.missing and na.rm.method). If no cases were removed the slot is NULL.
quarterly.vars	names of the quarterly variables (if any).
em.method	The EM method used.
call	call object obtained from match.call().

References

Doz, C., Giannone, D., & Reichlin, L. (2011). A two-step estimator for large approximate dynamic factor models based on Kalman filtering. *Journal of Econometrics, 164*(1), 188-205.

Doz, C., Giannone, D., & Reichlin, L. (2012). A quasi-maximum likelihood approach for large, approximate dynamic factor models. *Review of Economics and Statistics, 94*(4), 1014-1024.

Banbura, M., & Modugno, M. (2014). Maximum likelihood estimation of factor models on datasets with arbitrary pattern of missing data. *Journal of Applied Econometrics, 29*(1), 133-160.

Stock, J. H., & Watson, M. W. (2016). Dynamic Factor Models, Factor-Augmented Vector Autoregressions, and Structural Vector Autoregressions in Macroeconomics. *Handbook of Macroeconomics, 2*, 415–525. https://doi.org/10.1016/bs.hesmac.2016.04.002

See Also

dfms-package

Examples

```
library(magrittr)
library(xts)
library(vars)

# BM14 Replication Data. Constructing the database:
BM14 <- merge(BM14_M, BM14_Q)
BM14[, BM14_Models$log_trans] %<>% log()
BM14[, BM14_Models$freq == "M"] %<>% diff()
BM14[, BM14_Models$freq == "Q"] %<>% diff(3)


### Small Model -------------------------------------

# IC for number of factors
IC_small <- ICr(BM14[, BM14_Models$small], max.r = 5)
plot(IC_small)
screeplot(IC_small)

# I take 2 factors. Now number of lags
VARselect(IC_small$F_pca[, 1:2])

# Estimating the model with 2 factors and 3 lags
dfm_small <- DFM(BM14[, BM14_Models$small], r = 2, p = 3,
    quarterly.vars = BM14_Models %$% series[freq == "Q" & small])

# Inspecting the model
summary(dfm_small)
plot(dfm_small)  # Factors and data
plot(dfm_small, method = "all", type = "individual") # Factor estimates
plot(dfm_small, type = "residual") # Residuals from factor predictions

# 10 periods ahead forecast
plot(predict(dfm_small), xlim = c(300, 370))


### Medium-Sized Model ---------------------------------

# IC for number of factors
IC_medium <- ICr(BM14[, BM14_Models$medium])
plot(IC_medium)
screeplot(IC_medium)

# I take 3 factors. Now number of lags
VARselect(IC_medium$F_pca[, 1:3])

# Estimating the model with 3 factors and 3 lags
dfm_medium <- DFM(BM14[, BM14_Models$medium], r = 3, p = 3,
    quarterly.vars = BM14_Models %$% series[freq == "Q" & medium])

# Inspecting the model
summary(dfm_medium)
```

```
plot(dfm_medium)  # Factors and data
plot(dfm_medium, method = "all", type = "individual") # Factor estimates
plot(dfm_medium, type = "residual") # Residuals from factor predictions

# 10 periods ahead forecast
plot(predict(dfm_medium), xlim = c(300, 370))


### Large Model --------------------------------

# IC for number of factors
IC_large <- ICr(BM14)
plot(IC_large)
screeplot(IC_large)

# I take 6 factors. Now number of lags
VARselect(IC_large$F_pca[, 1:6])

# Estimating the model with 6 factors and 3 lags
dfm_large <- DFM(BM14, r = 6, p = 3,
    quarterly.vars = BM14_Models %$% series[freq == "Q"])

# Inspecting the model
summary(dfm_large)
plot(dfm_large)  # Factors and data
# plot(dfm_large, method = "all", type = "individual") # Factor estimates
plot(dfm_large, type = "residual") # Residuals from factor predictions

# 10 periods ahead forecast
plot(predict(dfm_large), xlim = c(300, 370))
```

---

em_converged	*Convergence Test for EM-Algorithm*

Description

Convergence Test for EM-Algorithm

Usage

```
em_converged(loglik, previous_loglik, tol = 1e-04, check.increased = FALSE)
```

Arguments

loglik   numeric. Current value of the log-likelihood function.

previous_loglik

                  numeric. Value of the log-likelihood function at the previous iteration.

tol   numerical. The tolerance of the test. If |LL(t) - LL(t-1)| / avg < tol, where avg = (|LL(t)| + |LL(t-1)|)/2, then algorithm has converged.

```
check.increased
```
>   logical. Check if likelihood has increased.

Value

A logical statement indicating whether EM algorithm has converged. if `check.increased = TRUE`, a vector with 2 elements indicating the convergence status and whether the likelihood has decreased.

See Also

[dfms-package](#)

Examples

```
em_converged(1001, 1000)
em_converged(10001, 10000)
em_converged(10001, 10000, check = TRUE)
em_converged(10000, 10001, check = TRUE)
```

FIS   *(Fast) Fixed-Interval Smoother (Kalman Smoother)*

Description

(Fast) Fixed-Interval Smoother (Kalman Smoother)

Usage

```
FIS(A, F, F_pred, P, P_pred, F_0 = NULL, P_0 = NULL)
```

Arguments

A	transition matrix ($rp \times rp$).
F	state estimates ($T \times rp$).
F_pred	state predicted estimates ($T \times rp$).
P	variance estimates ($rp \times rp \times T$).
P_pred	predicted variance estimates ($rp \times rp \times T$).
F_0	initial state vector ($rp \times 1$) or empty (NULL).
P_0	initial state covariance ($rp \times rp$) or empty (NULL).

Details

The Kalman Smoother is given by:

$$\mathbf{J}_t = \mathbf{P}_t\mathbf{A} + inv(\mathbf{P}_{t+1}^{pred})$$

$$\mathbf{F}_t^{smooth} = \mathbf{F}_t + \mathbf{J}_t(\mathbf{F}_{t+1}^{smooth} - \mathbf{F}_{t+1}^{pred})$$

$$\mathbf{P}_t^{smooth} = \mathbf{P}_t + \mathbf{J}_t(\mathbf{P}_{t+1}^{smooth} - \mathbf{P}_{t+1}^{pred})\mathbf{J}_t'$$

The initial smoothed values for period t = T are set equal to the filtered values. If F_0 and P_0 are supplied, the smoothed initial conditions (t = 0 values) are also calculated and returned. For further details see any textbook on time series such as Shumway & Stoffer (2017), which provide an analogous R implementation in astsa::Ksmooth0.

Value

Smoothed state and covariance estimates, including initial (t = 0) values.

F_smooth	$T \times rp$ smoothed state vectors, equal to the filtered state in period T.
P_smooth	$rp \times rp \times T$ smoothed state covariance, equal to the filtered covariance in period T.
F_smooth_0	$1 \times rp$ initial smoothed state vectors, based on F_0.
P_smooth_0	$rp \times rp$ initial smoothed state covariance, based on P_0.

References

Shumway, R. H., & Stoffer, D. S. (2017). Time Series Analysis and Its Applications: With R Examples. Springer.

Harvey, A. C. (1990). Forecasting, structural time series models and the Kalman filter.

See Also

SKF SKFS dfms-package

Examples

```
# See ?SKFS
```

ICr	*Information Criteria to Determine the Number of Factors (r)*

Description

Minimizes 3 information criteria proposed by Bai and Ng (2002) to determine the optimal number of factors r* to be used in an approximate factor model. A Screeplot can also be computed to eyeball the number of factors in the spirit of Onatski (2010).

Usage

```
ICr(X, max.r = min(20, ncol(X) - 1))

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

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

## S3 method for class 'ICr'
screeplot(x, type = "pve", show.grid = TRUE, max.r = 30, ...)
```

Arguments

X	a T x n numeric data matrix or frame of stationary time series.
max.r	integer. The maximum number of factors for which IC should be computed (or eigenvalues to be displayed in the screeplot).
x	an object of type 'ICr'.
...	further arguments to `ts.plot` or `plot`.
type	character. Either `"ev"` (eigenvalues), `"pve"` (percent variance explained), or `"cum.pve"` (cumulative PVE). Multiple plots can be requested.
show.grid	logical. TRUE shows gridlines in each plot.

Details

Following Bai and Ng (2002) and De Valk et al. (2019), let $NSSR(r)$ be the normalized sum of squared residuals $SSR(r)/(n \times T)$ when r factors are estimated using principal components. Then the information criteria can be written as follows:

$$IC_{r1} = \ln(NSSR(r)) + r\left(\frac{n+T}{nT}\right) + \ln\left(\frac{nT}{n+T}\right)$$

$$IC_{r2} = \ln(NSSR(r)) + r\left(\frac{n+T}{nT}\right) + \ln(\min(n,T))$$

$$IC_{r3} = \ln(NSSR(r)) + r\left(\frac{\ln(\min(n,T))}{\min(n,T)}\right)$$

The optimal number of factors r* corresponds to the minimum IC. The three criteria are are asymptotically equivalent, but may give significantly different results for finite samples. The penalty in IC_{r2} is highest in finite samples.

In the Screeplot a horizontal dashed line is shown signifying an eigenvalue of 1, or a share of variance corresponding to 1 divided by the number of eigenvalues.

Value

A list of 4 elements:

F_pca	T x n matrix of principle component factor estimates.
eigenvalues	the eigenvalues of the covariance matrix of X.
IC	r.max x 3 'table' containing the 3 information criteria of Bai and Ng (2002), computed for all values of r from 1:r.max.
r.star	vector of length 3 containing the number of factors (r) minimizing each information criterion.

Note

To determine the number of lags (p) in the factor transition equation, use the function vars::VARselect with r* principle components (also returned by ICr).

References

Bai, J., Ng, S. (2002). Determining the Number of Factors in Approximate Factor Models. *Econometrica, 70*(1), 191-221. https://doi.org/10.1111/1468-0262.00273.

Onatski, A. (2010). Determining the number of factors from empirical distribution of eigenvalues. *The Review of Economics and Statistics, 92*(4), 1004-1016.

De Valk, S., de Mattos, D., & Ferreira, P. (2019). Nowcasting: An R package for predicting economic variables using dynamic factor models. *The R Journal, 11*(1), 230-244.

See Also

dfms-package

Examples

```
library(xts)
library(vars)

ics <- ICr(diff(BM14_M))
print(ics)
plot(ics)
screeplot(ics)

# Optimal lag-order with 6 factors chosen
VARselect(ics$F_pca[, 1:6])
```

---

plot.dfm   *Plot DFM*

Description

Plot DFM

Usage

```
## S3 method for class 'dfm'
plot(
  x,
  method = switch(x$em.method, none = "2s", "qml"),
  type = c("joint", "individual", "residual"),
  scale.factors = TRUE,
  ...
)

## S3 method for class 'dfm'
screeplot(x, ...)
```

Arguments

x	an object class 'dfm'.
method	character. The factor estimates to use: one of "qml", "2s", "pca" or "all" to plot all estimates.
type	character. The type of plot: "joint", "individual" or "residual".
scale.factors	logical. Standardize factor estimates, this usually improves the plot since the factor estimates corresponding to the greatest PCA eigenvalues tend to have a greater variance than the data.
...	for plot.dfm: further arguments to [plot](#), [ts.plot](#), or [boxplot](#), depending on the type of plot. For screeplot.dfm: further arguments to [screeplot.ICr](#).

Value

Nothing.

See Also

[dfms-package](#)

Examples

```
# Fit DFM with 3 factors and 3 lags in the transition equation
mod <- DFM(diff(BM14_M), r = 3, p = 3)
plot(mod)
plot(mod, type = "individual", method = "all")
```

```
    plot(mod, type = "residual")
```

---

predict.dfm   *DFM Forecasts*

Description

This function produces h-step ahead forecasts of both the factors and the data, with an option to
also forecast autocorrelated residuals with a univariate method and produce a combined forecast.

Usage

```
## S3 method for class 'dfm'
predict(
  object,
  h = 10L,
  method = switch(object$em.method, none = "2s", "qml"),
  standardized = TRUE,
  resFUN = NULL,
  resAC = 0.1,
  ...
)

## S3 method for class 'dfm_forecast'
print(x, digits = 4L, ...)

## S3 method for class 'dfm_forecast'
plot(
  x,
  main = paste(x$h, "Period Ahead DFM Forecast"),
  xlab = "Time",
  ylab = "Standardized Data",
  factors = seq_len(ncol(x$F)),
  scale.factors = TRUE,
  factor.col = rainbow(length(factors)),
  factor.lwd = 1.5,
  fcst.lty = "dashed",
  data.col = c("grey85", "grey65"),
  legend = TRUE,
  legend.items = paste0("f", factors),
  grid = FALSE,
  vline = TRUE,
  vline.lty = "dotted",
  vline.col = "black",
  ...
```

```
)

## S3 method for class 'dfm_forecast'
as.data.frame(
  x,
  ...,
  use = c("factors", "data", "both"),
  pivot = c("long", "wide"),
  time = seq_len(nrow(x$F) + x$h),
  stringsAsFactors = TRUE
)
```

Arguments

object	an object of class 'dfm'.
h	integer. The forecast horizon.
method	character. The factor estimates to use: one of "qml", "2s" or "pca".
standardized	logical. FALSE will return data forecasts on the original scale.
resFUN	an (optional) function to compute a univariate forecast of the residuals. The function needs to have a second argument providing the forecast horizon (h) and return a vector of forecasts. See Examples.
resAC	numeric. Threshold for residual autocorrelation to apply resFUN: only residual series where AC1 > resAC will be forecasted.
...	not used.
x	an object class 'dfm_forecast'.
digits	integer. The number of digits to print out.
main, xlab, ylab	character. Graphical parameters passed to `ts.plot`.
factors	integers indicating which factors to display. Setting this to NA, NULL or 0 will omit factor plots.
scale.factors	logical. Standardize factor estimates, this usually improves the plot since the factor estimates corresponding to the greatest PCA eigenvalues tend to have a greater variance than the data.
factor.col, factor.lwd	
	graphical parameters affecting the colour and line width of factor estimates plots. See `par`.
fcst.lty	integer or character giving the line type of the forecasts of factors and data. See `par`.
data.col	character vector of length 2 indicating the colours of historical data and forecasts of that data. Setting this to NA, NULL or "" will not plot data and data forecasts.
legend	logical. TRUE draws a legend in the top-left of the chart.
legend.items	character names of factors for the legend.
grid	logical. TRUE draws a grid on the background of the plot.
vline	logical. TRUE draws a vertical line deliminating historical data and forecasts.

vline.lty, vline.col

        graphical parameters affecting the appearance of the vertical line. See [par](#).

use   character. Which forecasts to use "factors", "data" or "both".

pivot   character. The orientation of the frame: "long" or "wide".

time   a vector identifying the time dimension, must be of length T + h, or NULL to omit a time variable.

stringsAsFactors

        logical. If TRUE and pivot = "long" the 'Variable' column is created as a factor. Same as option to [as.data.frame.table](#).

Value

A list-like object of class 'dfm_forecast' with the following elements:

X_fcst $h \times n$ matrix with the forecasts of the variables.

F_fcst $h \times r$ matrix with the factor forecasts.

X $T \times n$ matrix with the standardized (scaled and centered) data - with attributes attached allowing reconstruction of the original data:

  "stats"   is a $n \times 5$ matrix of summary statistics of class "qsu" (see [qsu](#)). Only attached if standardized = TRUE.

  "attributes"   contains the [attributes](#) of the original data input.

  "is.list"   is a logical value indicating whether the original data input was a list / data frame.

  F $T \times r$ matrix of factor estimates.

  method   the factor estimation method used.

  anyNA   logical indicating whether X contains any missing values.

  h   the forecast horizon.

  resid.fc   logical indicating whether a univariate forecasting function was applied to the residuals.

  resid.fc.ind   indices indicating for which variables (columns of X) the residuals were forecasted using the univariate function.

  call   call object obtained from match.call().

See Also

[dfms-package](#)

Examples

```
library(xts)
library(collapse)

# Fit DFM with 3 factors and 3 lags in the transition equation
mod <- DFM(diff(BM14_M), r = 3, p = 3)

# 15 period ahead forecast
fc <- predict(mod, h = 15)
print(fc)
plot(fc, xlim = c(300, 370))

# Also forecasting autocorrelated residuals with an AR(1)
fcfun <- function(x, h) predict(ar(na_rm(x)), n.ahead = h)$pred
fcar <- predict(mod, resFUN = fcfun, h = 15)
plot(fcar, xlim = c(300, 370))

# Retrieving a data frame of the forecasts
head(as.data.frame(fcar, pivot = "wide")) # Factors
head(as.data.frame(fcar, use = "data"))   # Data
head(as.data.frame(fcar, use = "both"))   # Both
```

---

residuals.dfm *DFM Residuals and Fitted Values*

Description

The residuals $\mathbf{e}_t = \mathbf{x}_t - \mathbf{CF}_t$ or fitted values $\mathbf{CF}_t$ of the DFM observation equation.

Usage

```
## S3 method for class 'dfm'
residuals(
  object,
  method = switch(object$em.method, none = "2s", "qml"),
  orig.format = FALSE,
  standardized = FALSE,
  na.keep = TRUE,
  ...
)

## S3 method for class 'dfm'
fitted(
  object,
  method = switch(object$em.method, none = "2s", "qml"),
  orig.format = FALSE,
  standardized = FALSE,
```

```
  na.keep = TRUE,
  ...
)
```

Arguments

`object`	an object of class 'dfm'.
`method`	character. The factor estimates to use: one of "qml", "2s" or "pca".
`orig.format`	logical. TRUE returns residuals/fitted values in a data format similar to X.
`standardized`	logical. FALSE will put residuals/fitted values on the original data scale.
`na.keep`	logical. TRUE inserts missing values where X is missing (default TRUE as residuals/fitted values are only defined for observed data). FALSE returns the raw prediction, which can be used to interpolate data based on the DFM. For residuals, FALSE returns the difference between the prediction and the initial imputed version of X use for PCA to initialize the Kalman Filter.
`...`	not used.

Value

A matrix of DFM residuals or fitted values. If `orig.format = TRUE` the format may be different, e.g. a data frame.

See Also

[dfms-package](#)

Examples

```
library(xts)
# Fit DFM with 3 factors and 3 lags in the transition equation
mod <- DFM(diff(BM14_M), r = 3, p = 3)

# Residuals
head(resid(mod))
plot(resid(mod, orig.format = TRUE)) # this is an xts object

# Fitted values
head(fitted(mod))
head(fitted(mod, orig.format = TRUE)) # this is an xts object
```

SKF	*(Fast) Stationary Kalman Filter*

Description

A simple and fast C++ implementation of the Kalman Filter for stationary data (or random walks - data should be mean zero and without a trend) with time-invariant system matrices and missing data.

Usage

```
SKF(X, A, C, Q, R, F_0, P_0, loglik = FALSE)
```

Arguments

X	numeric data matrix $(T \times n)$.
A	transition matrix $(rp \times rp)$.
C	observation matrix $(n \times rp)$.
Q	state covariance $(rp \times rp)$.
R	observation covariance $(n \times n)$.
F_0	initial state vector $(rp \times 1)$.
P_0	initial state covariance $(rp \times rp)$.
loglik	logical. Compute log-likelihood?

Details

The underlying state space model is:

$$\mathbf{x}_t = \mathbf{CF}_t + \mathbf{e}_t \sim N(\mathbf{0}, \mathbf{R})$$

$$\mathbf{F}_t = \mathbf{A}\,\mathbf{F}_{t-1} + \mathbf{u}_t \sim N(\mathbf{0}, \mathbf{Q})$$

where x_t is X[t,]. The filter then first performs a time update (prediction)

$$\mathbf{F}_t = \mathbf{A}\,\mathbf{F}_{t-1}$$

$$\mathbf{P}_t = \mathbf{A}\,\mathbf{P}_{t-1}\mathbf{A}' + \mathbf{Q}$$

where $P_t = Cov(F_t)$. This is followed by the measurement update (filtering)

$$\mathbf{K}_t = \mathbf{P}_t\mathbf{C}'(\mathbf{C}\,\mathbf{P}_t\mathbf{C}' + \mathbf{R})^{-1}$$

$$\mathbf{F}_t = \mathbf{F}_t + \mathbf{K}_t(\mathbf{x}_t - \mathbf{C}\,\mathbf{F}_t)$$

$$\mathbf{P}_t = \mathbf{P}_t - \mathbf{K}_t\mathbf{C}\,\mathbf{P}_t$$

If a row of the data is all missing the measurement update is skipped i.e. the prediction becomes the filtered value. The log-likelihood is computed as

$$1/2 \sum_t \log(|St|) - e_t' S_t e_t - n \log(2\pi)$$

where $S_t = (CP_tC' + R)^{-1}$ and $e_t = x_t - CF_t$ is the prediction error.

For further details see any textbook on time series such as Shumway & Stoffer (2017), which provide an analogous R implementation in `astsa::Kfilter0`. For another fast (C-based) implementation that also allows time-varying system matrices and non-stationary data see `FKF::fkf`.

Value

Predicted and filtered state vectors and covariances.

F	$T \times rp$ filtered state vectors.
P	$rp \times rp \times T$ filtered state covariances.
F_pred	$T \times rp$ predicted state vectors.
P_pred	$rp \times rp \times T$ predicted state covariances.
loglik	value of the log likelihood.

References

Shumway, R. H., & Stoffer, D. S. (2017). Time Series Analysis and Its Applications: With R Examples. Springer.

Harvey, A. C. (1990). Forecasting, structural time series models and the Kalman filter.

Hamilton, J. D. (1994). Time Series Analysis. Princeton university press.

See Also

FIS SKFS dfms-package

Examples

```
# See ?SKFS
```

SKFS   *(Fast) Stationary Kalman Filter and Smoother*

Description

(Fast) Stationary Kalman Filter and Smoother

Usage

```
SKFS(X, A, C, Q, R, F_0, P_0, loglik = FALSE)
```

Arguments

X	numeric data matrix ($T \times n$).
A	transition matrix ($rp \times rp$).
C	observation matrix ($n \times rp$).
Q	state covariance ($rp \times rp$).
R	observation covariance ($n \times n$).
F_0	initial state vector ($rp \times 1$).
P_0	initial state covariance ($rp \times rp$).
loglik	logical. Compute log-likelihood?

Value

All results from [SKF](#) and [FIS](#), and additionally a $rp \times rp \times T$ matrix PPm_smooth, which is equal to the estimate of $Cov(F_t^{smooth}, F_{t-1}^{smooth}|T)$ and needed for EM iterations. See 'Property 6.3: The Lag-One Covariance Smoother' in Shumway & Stoffer (2017).

References

Shumway, R. H., & Stoffer, D. S. (2017). Time Series Analysis and Its Applications: With R Examples. Springer.

See Also

[SKF](#) [FIS](#) [dfms-package](#)

Examples

```
library(collapse)

## Two-Step factor estimates from monthly BM (2014) data
X <- fscale(diff(qM(BM14_M))) # Standardizing as KF has no intercept
r <- 5L # 5 Factors
p <- 3L # 3 Lags
n <- ncol(X)

## Initializing the Kalman Filter with PCA results
X_imp <- tsnarmimp(X)                   # Imputing Data
v <- eigen(cov(X_imp))$vectors[, 1:r] # PCA
F_pc <- X_imp %*% v                     # Principal component factor estimates
C <- cbind(v, matrix(0, n, r*p-r))    # Observation matrix
res <- X - tcrossprod(F_pc, v)        # Residuals from static predictions
R <- diag(fvar(res))                  # Observation residual covariance
var <- .VAR(F_pc, p)                  # VAR(p)
A <- rbind(t(var$A), diag(1, r*p-r, r*p))
Q <- matrix(0, r*p, r*p)              # VAR residual matrix
Q[1:r, 1:r] <- cov(var$res)
F_0 <- var$X[1L, ]                    # Initial factor estimate and covariance
P_0 <- ainv(diag((r*p)^2) - kronecker(A,A)) %*% unattrib(Q)
```

```
dim(P_0) <- c(r*p, r*p)

## Run standartized data through Kalman Filter and Smoother once
kfs_res <- SKFS(X, A, C, Q, R, F_0, P_0, FALSE)

## Two-step solution is state mean from the Kalman Smoother
F_kal <- kfs_res$F_smooth[, 1:r, drop = FALSE]
colnames(F_kal) <- paste0("f", 1:r)

## See that this is equal to the Two-Step estimate by DFM()
all.equal(F_kal, DFM(X, r, p, em.method = "none", pos.corr = FALSE)$F_2s)

## Same in two steps using SKF() and FIS()
kfs_res2 <- with(SKF(X, A, C, Q, R, F_0, P_0, FALSE), FIS(A, F, F_pred, P, P_pred))
F_kal2 <- kfs_res2$F_smooth[, 1:r, drop = FALSE]
colnames(F_kal2) <- paste0("f", 1:r)
all.equal(F_kal, F_kal2)

rm(X, r, p, n, X_imp, v, F_pc, C, res, R, var, A, Q, F_0, P_0, kfs_res, F_kal, kfs_res2, F_kal2)
```

---

summary.dfm   *DFM Summary Methods*

Description

Summary and print methods for class 'dfm'. print.dfm just prints basic model information and
the factor transition matrix **A**, coef.dfm returns **A** and **C** in a plain list, whereas summary.dfm re-
turns all system matrices and additional residual and goodness of fit statistics—with a print method
allowing full or compact printout.

Usage

```
## S3 method for class 'dfm'
print(x, digits = 4L, ...)

## S3 method for class 'dfm'
coef(object, ...)

## S3 method for class 'dfm'
logLik(object, ...)

## S3 method for class 'dfm'
summary(object, method = switch(object$em.method, none = "2s", "qml"), ...)

## S3 method for class 'dfm_summary'
print(x, digits = 4L, compact = sum(x$info["n"] > 15, x$info["n"] > 40), ...)
```

Arguments

`x`, `object`	an object class 'dfm'.
`digits`	integer. The number of digits to print out.
`...`	not used.
`method`	character. The factor estimates to use: one of `"qml"`, `"2s"` or `"pca"`.
`compact`	integer. Display a more compact printout: `0` prints everything, `1` omits the observation matrix **C** and residual covariance matrix `cov(resid(model))`, and `2` omits all disaggregated information on the input data. Sensible default are chosen for different sizes of the input dataset so as to limit large printouts.

Value

Summary information following a dynamic factor model estimation. `coef()` returns **A** and **C**.

See Also

[dfms-package](#)

Examples

```
mod <- DFM(diff(BM14_Q), 2, 3)
print(mod)
summary(mod)
```

---

tsnarmimp	*Remove and Impute Missing Values in a Multivariate Time Series*

Description

This function imputes missing values in a stationary multivariate time series using various methods, and removes cases with too many missing values.

Usage

```
tsnarmimp(
  X,
  max.missing = 0.8,
  na.rm.method = c("LE", "all"),
  na.impute = c("median.ma.spline", "median.ma", "median", "rnorm"),
  ma.terms = 3L
)
```

Arguments

X	a T x n numeric data matrix (incl. ts or xts objects) or data frame of stationary time series.
max.missing	numeric. Proportion of series missing for a case to be considered missing.
na.rm.method	character. Method to apply concerning missing cases selected through max.missing: ″LE″ only removes cases at the beginning or end of the sample, whereas ″all″ always removes missing cases.
na.impute	character. Method to impute missing values for the PCA estimates used to initialize the EM algorithm. Note that data are standardized (scaled and centered) beforehand. Available options are:

″median″	simple series-wise median imputation.
″rnorm″	imputation with random numbers drawn from a standard normal distribution.
″median.ma″	values are initially imputed with the median, but then a moving average is applied to smooth the e
″median.ma.spline″	″internal″ missing values (not at the beginning or end of the sample) are imputed using a cubic sp

ma.terms	the order of the (2-sided) moving average applied in na.impute methods ″median.ma″ and ″median.ma.spline″.

Value

The imputed matrix X_imp, with attributes:

″missing″	a missingness matrix W matching the dimensions of X_imp.
″rm.rows″	and a vector of indices of rows (cases) with too many missing values that were removed.

See Also

dfms-package

Examples

```
library(xts)
str(tsnarmimp(BM14_M))
```

Index