Package ‘bayesdfa’

May 22, 2019

Type Package

Title Bayesian Dynamic Factor Analysis (DFA) with 'Stan'

Version 0.1.3

Description Implements Bayesian dynamic factor analysis with 'Stan'. Dynamic factor analysis is a dimension reduction tool for multivariate time series. ‘bayesdfa’ extends conventional dynamic factor models in several ways. First, extreme events may be estimated in the latent trend by modeling process error with a student-t distribution. Second, autoregressive and moving average components can be optionally included. Third, the estimated dynamic factors can be analyzed with hidden Markov models to evaluate support for latent regimes.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Depends R (>= 3.4.0), Rcpp (>= 0.12.18), methods

Imports rstan (>= 2.18.2), rstantools (>= 1.5.1), ggplot2, loo (>= 2.0.0), dplyr (>= 0.8.0), reshape2, rlang (>= 0.3.1)

LinkingTo StanHeaders (>= 2.18.1), rstan (>= 2.18.2), BH (>= 1.66.0), Rcpp (>= 0.12.8), RcppEigen (>= 0.3.3.3.0)

Suggests testthat, parallel, knitr, rmarkdown, MARSS

URL https://github.com/fate-ewi/bayesdfa

BugReports https://github.com/fate-ewi/bayesdfa/issues

RoxygenNote 6.1.1

VignetteBuilder knitr

SystemRequirements GNU make

NeedsCompilation yes

Author Eric J. Ward [aut, cre], Sean C. Anderson [aut], Luis A. Damiano [aut], Mary E. Hunsicker, [ctb],
The 'bayesdfa' package.

A description of the package

References

find_dfa_trends

Find the best number of trends according to LOOIC

Description

Fit a DFA with different number of trends and return the leave one out (LOO) value as calculated by the loo package.

Usage

find_dfa_trends(y = y, kmin = 1, kmax = 5, iter = 2000, thin = 1,
                compare_normal = FALSE, convergence_threshold = 1.05,
                variance = c("equal", "unequal"), ...)

Arguments

y            A matrix of data to fit. Columns represent time element.
kmin         Minimum number of trends, defaults to 1.
kmax         Maximum number of trends, defaults to 5.
iter          Iterations when sampling from each Stan model, defaults to 2000.
thin          Thinning rate when sampling from each Stan model, defaults to 1.
compare_normal If TRUE, does model selection comparison of Normal vs. Student-t errors
convergence_threshold
                The maximum allowed value of Rhat to determine convergence of parameters
variance      Vector of variance arguments for searching over large groups of models. Can be either or both of ("equal", "unequal")
...            Other arguments to pass to fit_dfa()

Examples

set.seed(42)
s <- sim_dfa(num_trends = 2, num_years = 20, num_ts = 3)
# only 1 chain and 180 iterations used so example runs quickly:
m <- find_dfa_trends(
  y = s$y_sim, iter = 180,
  kmin = 1, kmax = 2, chains = 1, compare_normal = FALSE,
  variance = "equal", convergence_threshold = 1.1,
  control = list(adapt_delta = 0.95, max_treedepth = 20))
m$summary
m$best_model
find_inverted_chains  Find which chains to invert

Description

Find which chains to invert by checking the sum of the squared deviations between the first chain and each other chain.

Usage

find_inverted_chains(model, trend = 1, plot = FALSE)

Arguments

model              A Stan model, rstanfit object
trend              Which trend to check
plot                Logical: should a plot of the trend for each chain be made? Defaults to FALSE

See Also

invert_chains

Examples

set.seed(2)
s <- sim_dfa(num_trends = 2)
set.seed(1)
m <- fit_dfa(y = s$y_sim, num_trends = 1, iter = 500, chains = 2)
# chains were already inverted, but we can redo that, as an example, with:
find_inverted_chains(m$model, plot = TRUE)

find_regimes  Fit multiple models with differing numbers of regimes to trend data

Description

Fit multiple models with differing numbers of regimes to trend data

Usage

find_regimes(y, sds = NULL, min_regimes = 1, max_regimes = 3, iter = 2000, thin = 1, chains = 1, ...)

find_swans

Arguments

- **y**: Data, time series or trend from fitted DFA model.
- **sds**: Optional time series of standard deviations of estimates. If passed in, residual variance not estimated.
- **min_regimes**: Smallest of regimes to evaluate, defaults to 1.
- **max_regimes**: Biggest of regimes to evaluate, defaults to 3.
- **iter**: MCMC iterations, defaults to 2000.
- **thin**: MCMC thinning rate, defaults to 1.
- **chains**: MCMC chains; defaults to 1 (note that running multiple chains may result in a "label switching" problem where the regimes are identified with different IDs across chains).
- **...**: Other parameters to pass to `rstan::sampling()`.

Examples

```r
data(Nile)
find_regimes(log(Nile), iter = 500, chains = 1, max_regimes = 2)
```

---

find_swans

*Find outlying "black swan" jumps in trends*

Description

Find outlying "black swan" jumps in trends

Usage

```r
find_swans(rotated_modelfit, threshold = 0.01, plot = FALSE)
```

Arguments

- **rotated_modelfit**: Output from `rotate_trends()`.
- **threshold**: A probability threshold below which to flag trend events as extreme
- **plot**: Logical: should a plot be made?

Value

Prints a ggplot2 plot if `plot = TRUE`; returns a data frame indicating the probability that any given point in time represents a "black swan" event invisibly.

References

Examples

set.seed(1)
s <- sim_dfa(num_trends = 1, num_ts = 3, num_years = 30)
s$y_sim[1, 15] <- s$y_sim[1, 15] - 6
plot(s$y_sim[,], type = "o")
abline(v = 15, col = "red")
# only 1 chain and 250 iterations used so example runs quickly:
m <- fit_dfa(y = s$y_sim, num_trends = 1, iter = 250, chains = 1, nu_fixed = 2)
r <- rotate_trends(m)
p <- plot_trends(r) #+ geom_vline(xintercept = 15, colour = "red")
print(p)
# a 1 in 1000 probability if was from a normal distribution:
find_swans(r, plot = TRUE, threshold = 0.001)

---

fit_dfa

Fit a Bayesian DFA

Description

Fit a Bayesian DFA

Usage

`fit_dfa(y = y, num_trends = 1, varindx = NULL, zscore = TRUE, iter = 2000, chains = 4, thin = 1, control = list(adapt_delta = 0.99, max_treedepth = 20), nu_fixed = 101, est_correlation = FALSE, estimate_nu = FALSE, estimate_trend_ar = FALSE, estimate_trend_ma = FALSE, sample = TRUE, data_shape = c("wide", "long"), obs_covar = NULL, pro_covar = NULL, ...)`

Arguments

- `y`: A matrix of data to fit. See `data_shape` option to specify whether this is long or wide format data.
- `num_trends`: Number of trends to fit.
- `varindx`: Indices indicating which timeseries should have shared variances.
- `zscore`: Logical. Should the data be standardized first? If not it is just centered. Centering is necessary because no intercept is included.
- `iter`: Number of iterations in Stan sampling, defaults to 2000.
- `chains`: Number of chains in Stan sampling, defaults to 4.
- `thin`: Thinning rate in Stan sampling, defaults to 1.
- `control`: A list of options to pass to Stan sampling. Defaults to `list(adapt_delta = 0.99, max_treedepth = 20)`, `nu_fixed = 101`, `est_correlation = FALSE`, `estimate_nu = FALSE`, `estimate_trend_ar = FALSE`, `estimate_trend_ma = FALSE`, `sample = TRUE`, `data_shape = c("wide", "long"), obs_covar = NULL, pro_covar = NULL, ...`
est_correlation

Boolean, whether to estimate correlation of observation error matrix \( \mathbf{r} \). Defaults to FALSE.

estimate_nu

Logical. Estimate the student t degrees of freedom parameter? Defaults to FALSE.

estimate_trend_ar

Logical. Estimate AR(1) parameters on DFA trends? Defaults to `FALSE`, in which case AR(1) parameters are set to 1.

estimate_trend_ma

Logical. Estimate MA(1) parameters on DFA trends? Defaults to `FALSE`, in which case MA(1) parameters are set to 0.

sample

Logical. Should the model be sampled from? If FALSE, then the data list object that would have been passed to Stan is returned instead. This is useful for debugging and simulation. Defaults to TRUE.

data_shape

If wide (the current default) then the input data should have rows representing the various timeseries and columns representing the values through time. This matches the MARSS input data format. If long then the input data should have columns representing the various timeseries and rows representing the values through time.

obs_covar

Optional dataframe of data with 4 named columns ("time","timeseries","covariate","value"), representing: (1) time, (2) the time series affected, (3) the covariate number for models with more than one covariate affecting each trend, and (4) the value of the covariate

pro_covar

Optional dataframe of data with 4 named columns ("time","trend","covariate","value"), representing: (1) time, (2) the trend affected, (3) the covariate number for models with more than one covariate affecting each trend, and (4) the value of the covariate

... Any other arguments to pass to rstan::sampling().

Details

Note that there is nothing restricting the loadings and trends from being inverted (i.e. multiplied by \(-1\)) for a given chain. Therefore, if you fit multiple chains, the package will attempt to determine which chains need to be inverted using the function find_inverted_chains().

See Also

plot_loadings plot_trends rotate_trends find_swans

Examples

```r
set.seed(42)
s <- sim_dfa(num_trends = 1, num_years = 20, num_ts = 3)
# only 1 chain and 250 iterations used so example runs quickly:
m <- fit_dfa(y = s$y_sim, iter = 250, chains = 1)
## Not run:
# example of observation error covariates
obs_covar = expand.grid("time"=1:20,"timeseries"=1:3,"covariate"=1)
```
Fit models with differing numbers of regimes to trend data

Description

Fit models with differing numbers of regimes to trend data

Usage

```r
fit_regimes(y, sds = NULL, n_regimes = 2, iter = 2000, thin = 1,
            chains = 1, ...)```

Arguments

- `y` Data, time series or trend from fitted DFA model.
- `sds` Optional time series of standard deviations of estimates. If passed in, residual variance not estimated. Defaults to `NULL`.
- `n_regimes` Number of regimes to evaluate, defaults 2.
- `iter` MCMC iterations, defaults to 2000.
- `thin` MCMC thinning rate, defaults to 1.
- `chains` MCMC chains, defaults to 1 (note that running multiple chains may result in a label switching problem where the regimes are identified with different IDs across chains).
- `...` Other parameters to pass to `rstan::sampling()`.

Examples

```r
data(Nile)
fit_regimes(log(Nile), iter = 1000, n_regimes = 1)
```
hmm_init

Create initial values for the HMM model.

Description
Create initial values for the HMM model.

Usage
hmm_init(K, x_t)

Arguments
K The number of regimes or clusters to fit. Called by rstan::sampling().
x_t A matrix of values. Called by rstan::sampling().

Value
list of initial values (mu, sigma)

invert_chains

Invert chains

Description
Invert chains

Usage
invert_chains(model, trends = 1, print = FALSE, ...)

Arguments
model A Stan model, rstanfit object
trends The number of trends in the DFA, defaults to 1
print Logical indicating whether the summary should be printed. Defaults to FALSE.
... Other arguments to pass to find_inverted_chains().

See Also
find_inverted_chains
is_converged  

_Summarize Rhat convergence statistics across parameters_

**Description**  
Pass in `rstanfit` model object, and a threshold Rhat value for convergence. Returns boolean.

**Usage**  
is_converged(fitted_model, threshold = 1.05, parameters = c("sigma", "x", "Z"))

**Arguments**  
fitted_model  
Samples extracted (with `permuted = FALSE`) from a Stan model. E.g. output from `invert_chains()`.

threshold  
Threshold for maximum Rhat.

parameters  
Vector of parameters to be included in convergence determination. Defaults = c("sigma","x","Z"). Other elements can be added including "pred", "log_lik", or "lp__"

---

loo.bayesdfa  

_LOO information criteria_

**Description**  
Extract the LOOIC (leave-one-out information criterion) using `loo::loo()`. Note that we’ve implemented slightly different variants of loo, based on whether the DFA observation model includes correlation between time series or not (default is no correlation). Importantly, these different versions are not directly comparable to evaluate data support for including correlation or not in a DFA. If time series are not correlated, the point-wise log-likelihood for each observation is calculated and used in the loo calculations. However if time series are correlated, then each time slice is assumed to be a joint observation of all variables, and the point-wise log-likelihood is calculated as the joint likelihood of all variables under the multivariate normal distribution.

**Usage**  
```r  
## S3 method for class 'bayesdfa'  
loo(x, ...)  
```

**Arguments**  
x  
Output from `fit_dfa()`.

...  
Arguments for `loo::relative_eff()` and `loo::loo.array()`.
Examples

```r
set.seed(1)
s <- sim_dfa(num_trends = 1, num_years = 50, num_ts = 3)
m <- fit_dfa(y = s$y_sim, iter = 300, chains = 1, num_trends = 1)
loo(m)
```

---

**plot_fitted**  
*Plot the trends from a DFA*

**Description**

Plot the trends from a DFA

**Usage**

```r
plot_fitted(modelfit, names = NULL)
```

**Arguments**

- `modelfit`: Output from `fit_dfa`, a rstanfit object
- `names`: Optional vector of names for plotting labels

**See Also**

- `plot_loadings`
- `fit_dfa`
- `rotate_trends`

**Examples**

```r
y <- sim_dfa(num_trends = 2, num_years = 20, num_ts = 4)
m <- fit_dfa(y = y, num_trends = 2, iter = 200, chains = 1)
p <- plot_fitted(m)
print(p)
```
plot_loadings

Plot the loadings from a DFA

Description
Plot the loadings from a DFA

Usage

plot_loadings(rotated_modelfit, names = NULL, facet = TRUE, violin = TRUE, conf_level = 0.95, threshold = NULL)

Arguments

rotated_modelfit
Output from rotate_trends().

names
An optional vector of names for plotting the loadings.

facet
Logical. Should there be a separate facet for each trend? Defaults to TRUE.

violin
Logical. Should the full posterior densities be shown as a violin plot? Defaults to TRUE.

conf_level
Confidence level for credible intervals. Defaults to 0.95.

threshold
Numeric (0-1). Optional for plots, if included, only plot loadings who have Pr(<0) or Pr(>0) > threshold. For example threshold = 0.8 would only display estimates where 80 zero. Defaults to NULL (not used).

See Also
plot_trends fit_dfa rotate_trends

Examples

set.seed(42)
s <- sim_dfa(num_trends = 2, num_ts = 4, num_years = 10)
# only 1 chain and 180 iterations used so example runs quickly:
m <- fit_dfa(y = s$y_sim, num_trends = 2, iter = 180, chains = 1)
r <- rotate_trends(m)
plot_loadings(r, violin = FALSE, facet = TRUE)
plot_loadings(r, violin = FALSE, facet = FALSE)
plot_loadings(r, violin = TRUE, facet = FALSE)
plot_loadings(r, violin = TRUE, facet = TRUE)
plot_regime_model

Plot the state probabilities from find_regimes()

Description

Plot the state probabilities from find_regimes()

Usage

plot_regime_model(model, probs = c(0.05, 0.95), type = c("probability", "means"), regime_prob_threshold = 0.9, plot_prob_indices = NULL, flip_regimes = FALSE)

Arguments

model A model returned by find_regimes().
probs A numeric vector of quantiles to plot the credible intervals at. Defaults to c(0.05, 0.95).
type Whether to plot the probabilities (default) or means.
regime_prob_threshold The probability density that must be above 0.5. Defaults to 0.9 before we classify a regime (only affects "means" plot).
plot_prob_indices Optional indices of probability plots to plot. Defaults to showing all.
flip_regimes Optional whether to flip regimes in plots, defaults to FALSE

Details

Note that the original timeseries data (dots) are shown scaled between 0 and 1.

Examples

data(Nile)
m <- fit_regimes(log(Nile), n_regimes = 2, chains = 1, iter = 800)
plot_regime_model(m)
plot_regime_model(m, plot_prob_indices=c(2))
plot_regime_model(m, type = "means")
plot_trends

Plot the trends from a DFA

Description

Plot the trends from a DFA

Usage

plot_trends(rotated_modelfit, years = NULL, highlight_outliers = FALSE, threshold = 0.01)

Arguments

- rotated_modelfit: Output from `rotate_trends`
- years: Optional numeric vector of years for the plot
- highlight_outliers: Logical. Should trend events that exceed the probability of occurring with a normal distribution as defined by `threshold` be highlighted? Defaults to FALSE
- threshold: A probability threshold below which to flag trend events as extreme. Defaults to 0.01

See Also

plot_loadings fit_dfa rotate_trends

Examples

```r
set.seed(1)
s <- sim_dfa(num_trends = 1)
m <- fit_dfa(y = s$y_sim, num_trends = 1, iter = 500, chains = 1)
r <- rotate_trends(m)
p <- plot_trends(r)
print(p)
```

predicted

Calculate predicted value from DFA object

Description

Pass in rstanfit model object. Returns array of predictions, dimensioned number of MCMC draws x number of MCMC chains x time series length x number of time series
rotate_trends

Usage

predicted(fitted_model)

Arguments

fitted_model Samples extracted (with permuted = FALSE) from a Stan model. E.g. output from `invert_chains()`.

Examples

```r
set.seed(42)
s <- sim_dfa(num_trends = 1, num_years = 20, num_ts = 3)
# only 1 chain and 1000 iterations used so example runs quickly:
m <- fit_dfa(y = s$y_sim, iter = 1000, chains = 1)
pred <- predicted(m)
```

---

rotate_trends Rotate the trends from a DFA

Description

Rotate the trends from a DFA

Usage

rotate_trends(fitted_model, conf_level = 0.95, invert = FALSE)

Arguments

fitted_model Output from `fit_dfa()`.
conf_level Probability level for CI.
invert Whether to invert the trends and loadings for plotting purposes

Examples

```r
set.seed(42)
s <- sim_dfa(num_trends = 1, num_years = 20, num_ts = 3)
# only 1 chain and 800 iterations used so example runs quickly:
m <- fit_dfa(y = s$y_sim, iter = 800, chains = 1)
r <- rotate_trends(m)
plot_trends(r)
```
**sim_dfa**

*Simulate from a DFA*

**Description**

Simulate from a DFA

**Usage**

```r
sim_dfa(num_trends = 1, num_years = 20, num_ts = 4,
loadings_matrix = matrix(nrow = num_ts, ncol = num_trends, rnorm(num_ts
* num_trends, 0, 1)), sigma = rlnorm(1, meanlog = log(0.2), 0.1),
varIndx = rep(1, num_ts), extreme_value = NULL, extreme_loc = NULL,
nu_fixed = 100, user_supplied_deviations = NULL)
```

**Arguments**

- `num_trends` The number of trends.
- `num_years` The number of years.
- `num_ts` The number of timeseries.
- `loadings_matrix` A loadings matrix. The number of rows should match the number of timeseries and the number of columns should match the number of trends. Note that this loadings matrix will be internally manipulated by setting some elements to 0 and constraining some elements to 1 so that the model can be fitted. See `fit_dfa()`.
  - `z` in the returned list is to see the manipulated loadings matrix. If not specified, a random matrix ~ N(0, 1) is used.
- `sigma` A vector of standard deviations on the observation error. Should be of the same length as the number of trends. If not specified, random numbers are used `rlnorm(1, meanlog = log(0.2), 0.1)`.
- `varIndx` Indices of unique observation variances. Defaults to c(1, 1, 1, 1). Unique observation error variances would be specified as c(1, 2, 3, 4) in the case of 4 time series.
- `extreme_value` Value added to the random walk in the extreme time step. Defaults to not included.
- `extreme_loc` Location of single extreme event in the process. The same for all processes, and defaults to `round(n_t/2)` where n_t is the time series length.
- `nu_fixed` Nu is the degrees of freedom parameter for the t-distribution, defaults to 100, which is effectively normal.
- `user_supplied_deviations` An optional matrix of deviations for the trend random walks. Columns are for trends and rows are for each time step.
trend_cor

Value

A list with the following elements: y_sim is the simulated data, pred is the true underlying data without observation error added, x is the underlying trends, Z is the manipulated loadings matrix that is fed to the model.

Examples

```r
x <- sim_dfa(num_trends = 2)
names(x)
matplot(t(x$y_sim), type = "l")
matplot(t(x$x), type = "l")

set.seed(42)
x <- sim_dfa(extreme_value = -4, extreme_loc = 10)
matplot(t(x$x), type = "l"); abline(v = 10)
matplot(t(x$pred), type = "l"); abline(v = 10)

set.seed(42)
x <- sim_dfa()
matplot(t(x$x), type = "l"); abline(v = 10)
matplot(t(x$pred), type = "l"); abline(v = 10)
```

trend_cor

Estimate the correlation between a DFA trend and some other time-series

Description

Fully incorporates the uncertainty from the posterior of the DFA trend

Usage

trend_cor(rotated_modelfit, y, trend = 1,
  time_window = seq_len(length(y)), trend_samples = 100,
  stan_iter = 300, stan_chains = 1, ...)

Arguments

- **rotated_modelfit** Output from `rotate_trends()`.
- **y** A numeric vector to correlate with the DFA trend. Must be the same length as the DFA trend.
- **trend** A number corresponding to which trend to use, defaults to 1.
- **time_window** Indices indicating a time window slice to use in the correlation. Defaults to using the entire time window. Can be used to walk through the timeseries and test the cross correlations.
trend_samples  The number of samples from the trend posterior to use. A model will be run for each trend sample so this value shouldn’t be too large. Defaults to 100.

stan_iter  The number of samples from the posterior with each Stan model run, defaults to 300.

stan_chains  The number of chains for each Stan model run, defaults to 1.

...  Other arguments to pass to `sampling`

**Details**

Uses a $\sigma \sim \text{half}_t(3, 0, 2)$ prior on the residual standard deviation and a $\text{uniform}(-1, 1)$ prior on the correlation coefficient. Fitted as a linear regression of $y \sim x$, where $y$ represents the $y$ argument to `trend_cor()` and $x$ represents the DFA trend, and both $y$ and $x$ have been scaled by subtracting their means and dividing by their standard deviations. Samples are drawn from the posterior of the trend and repeatedly fed through the Stan regression to come up with a combined posterior of the correlation.

**Value**

A numeric vector of samples from the correlation coefficient posterior.

**Examples**

```r
set.seed(1)
s <- sim_dfa(num_trends = 1, num_years = 15)
m <- fit_dfa(y = s$y_sim, num_trends = 1, iter = 500, chains = 1)
r <- rotate_trends(m)
n_years <- ncol(r$trends[,1,])
fake_dat <- rnorm(n_years, 0, 1)
correlation <- trend_cor(r, fake_dat, trend_samples = 25)
hist(correlation)
correlation <- trend_cor(r, y = fake_dat, time_window = 5:15,
  trend_samples = 25)
hist(correlation)
```
Index

bayesdfa (bayesdfa-package), 2
bayesdfa-package, 2

find_dfa_trends, 3
find_inverted_chains, 4
find_inverted_chains(), 7, 9
find_regimes, 4
find_regimes(), 13
find_swans, 5
fit_dfa, 6, 11
fit_dfa(), 10, 15, 16
fit_regimes, 8

hmm_init, 9

invert_chains, 9
invert_chains(), 10, 15
is_converged, 10

loo, 3
loo (loo.bayesdfa), 10
loo.bayesdfa, 10
loo::loo(), 10
loo::loo.array(), 10
loo::relative_eff(), 10

plot_fitted, 11
plot_loadings, 12
plot_regime_model, 13
plot_trends, 14
predicted, 14

rotate_trends, 14, 15
rotate_trends(), 5, 12, 17
rstan::sampling(), 5, 7–9

sampling, 18
sim_dfa, 16

trend_cor, 17
trend_cor(), 18