Package ‘fosr’

August 5, 2018

Type Package

Title

Version 1.0

Date 2018-04-12

Author Daniel Bourgeois

Maintainer Daniel Bourgeois <dcb10@rice.edu>

Description

Imports Rcpp (>= 0.12.14), gibber, dsp, truncdist, MASS

License What license is it under?

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 6.0.1

R topics documented:

fosr *MCMC Sampling Algorithm for the Function-on-Scalars Regression Model*

Description

Runs the MCMC for the function-on-scalars regression model based on an FDLM-type expansion. Here we assume the factor regression has independent errors, which allows for subject-specific random effects, as well as some additional default conditions.

Usage

```
fosr(Y, tau, X = NULL, K = NULL, nsave = 1000, nburn = 1000,
  nskip = 3, mcmc_params = list("beta", "fk", "alpha", "sigma_e",
  "sigma_g"), computeDIC = TRUE)
```

Arguments

`Y`	the `n x m` data observation matrix, where `n` is the number of subjects and `m` is the number of observation points (`NA`s allowed)
`tau`	the `m x d` matrix of coordinates of observation points
`X`	the `n x p` matrix of predictors; if NULL, only include an intercept
`K`	the number of factors; if NULL, use SVD-based proportion of variability explained
`nsave`	number of MCMC iterations to record
`nburn`	number of MCMC iterations to discard (burin-in)
`nskip`	number of MCMC iterations to skip between saving iterations, i.e., save every (nskip + 1)th draw
`mcmc_params`	named list of parameters for which we store the MCMC output; must be one or more of • "beta" (factors) • "fk" (loading curves) • "alpha" (regression coefficients) • "mu_k" (intercept term for factor k) • "sigma_e" (observation error SD) • "sigma_g" (random effects SD) • "Yhat" (fitted values)
`computeDIC`	logical; if TRUE, compute the deviance information criterion `DIC` and the effective number of parameters `p_d`

Value

A named list of the `nsave` MCMC samples for the parameters named in `mcmc_params`

Note

If nm is large, then storing all posterior samples for Yhat, which is nsave x n x M, may be inefficient

Examples

```
# Simulate some data:
sim_data = simulate_fosr(n = 100, m = 20, p_0 = 100, p_1 = 5)

# Data:
Y = sim_data$Y; X = sim_data$X; tau = sim_data$tau

# Dimensions:
n = nrow(Y); m = ncol(Y); p = ncol(X)

# Run the FOSR:
out = fosr(Y = Y, tau = tau, X = X, K = 6, mcmc_params = list("fk", "alpha", "Yhat"))

# Plot a posterior summary of a regression function, say j = 3:
j = 3; post_alpha_tilde_j = get_post_alpha_tilde(out$fk, out$alpha[,j,])
plot_curve(post_alpha_tilde_j, tau = tau)
# Add the true curve:
lines(tau, sim_data$alpha_tilde_true[,j], lwd=6, col='green', lty=6)

# Plot the loading curves:
plot_flc(out$fk, tau = tau)

# Plot the fitted values for a random subject:
i = sample(1:n, 1)
plot_fitted(y = Y[i,], mu = colMeans(out$Yhat[,i,]),
            postY = out$Yhat[,i,], y_true = sim_data$Y_true[i,], t01 = tau)
```

---

fosr_gbpv *Compute Global Bayesian P-Values*

Description

Given posterior samples for the loading curves fk and the regression coefficient factors alpha, compute Global Bayesian P-Values for all regression coefficient functions

Usage

```
fosr_gbpv(post_fk, post_alpha)
```

Arguments

post_fk	Nsims x m x K matrix of posterior draws of the loading curve matrix
post_alpha_j	Nsims x p x K matrix of posterior draws of the regression coefficient factors

Value

p x 1 vector of Global Bayesian P-Values

---

fosr_select *Decoupling shrinkage and selection for function-on-scalars regression*

Description

For a functional response and scalar predictors, construct a posterior summary that balances predictive accuracy and sparsity. Given posterior draws of regression coefficients (or coefficient functions) from a FOSR model, use a suitably-defined loss function to select important variables for prediction.

Usage

```
fosr_select(X, post_alpha, post_trace_sigma_2, weighted = TRUE,
  alpha_level = 0.1, remove_int = TRUE, include_plot = TRUE,
  include_model_list = FALSE)
```

Arguments

`X`	`n x p` matrix of predictors
`post_alpha`	`Nsims x p x K` array of `Nsims` posterior draws of the `p` predictors for each of `K` factors
`post_trace_sigma_2`	`Nsims x 1` vector of posterior draws of the trace of the (marginal) covariance (see below for details)
`weighted`	logical; if TRUE, use weighted group lasso (recommended)
`alpha_level`	coverage for the credible interval on the proportion of variance explained
`remove_int`	logical; if TRUE, remove the intercept term from model comparisons
`include_plot`	logical; if TRUE, include a plot showing proportion of variability explained against model size
`include_model_list;`	if TRUE, include model_list in return–a boolean matrix of models of different sizes suggested by DSS

Value

alpha_dss a `p x K` matrix of (sparse) regression coefficents; if include_model_list is TRUE, return a list of alpha_dss and model_list, a boolean matrix of possible different sized models

Note

This function is value for the regression functions (m-dimensional) as well as the regression factors (K-dimensional). Since K « m, the latter is much faster.

The matrix of predictors, X, may be different from the given matrix in the data; i.e., we may have a different set of design points for prediction.

post_trace_sigma_2 is the (posterior samples of) the trace of the error covariance matrix jointly across subjects i=1,...,n and observations j=1,...,m, after marginalizing out the random effects gamma_ik. This is given by nm x sigma_e^2 + sum_ik sigma_gamma_ik^2, where the second term is necessary only when random effects are included in the model AND integrated over in the predictive distribution.

Examples

```
# Simulate some data:
sim_data = simulate_fosr(n = 100, m = 20, p_0 = 100, p_1 = 5)

# Data:
Y = sim_data$Y; X = sim_data$X; tau = sim_data$tau

# Dimensions:
n = nrow(Y); m = ncol(Y); p = ncol(X)

# Run the FOSR:
out = fosr(Y = Y, tau = tau, X = X, K = 6, mcmc_params = list("fk", "alpha", "Yhat"))

# Run the DSS:
alpha_dss = fosr_select(X = X,
                       post_alpha = out$alpha,
                   post_trace_sigma_2 = n*m*out$sigma_e^2 + apply(out$sigma_g^2, 1, sum))
# Variables selected:
(select_dss = which(apply(alpha_dss, 1, function(x) any(x != 0))))
```

---

get_post_alpha_tilde *Compute the posterior distrubution for the regression coefficient functions*

Description

Given posterior samples for the loading curves fk and the regression coefficient factors alpha_j for a predictor j, compute the posterior distribution of the corresponding regression coefficient function.

Usage

```
get_post_alpha_tilde(post_fk, post_alpha_j)
```

Arguments

`post_fk`	`Nsims x m x K` matrix of posterior draws of the loading curve matrix
`post_alpha_j`	`Nsims x K` matrix of posterior draws of the regression coefficient factors

Value

`Nsims x m` matrix of posterior draws of the regression coefficient function

---

`plot_curve` *Plot a curve given posterior samples*

Description

Plot the posterior mean, simultaneous and pointwise 95% credible bands for a curve given draws from the posterior distribution

Usage

```
plot_curve(post_f, tau = NULL, alpha = 0.05, include_joint = TRUE,
  main = "Posterior Mean and Credible Bands")
```

Arguments

`post_f`	`Ns x m` matrix of `Ns` posterior simulations of the curve at `m` points
`tau`	`m x 1` vector of observation points
`alpha`	confidence level for the bands
`include_joint`	logical; if TRUE, include joint bands (as well as pointwise)
`main`	text for title plot

`plot_factors` *Plot the factors*

Description

Plot posterior mean of the factors together with the simultaneous and pointwise 95% credible bands.

Usage

```
plot_factors(post_beta, subj = NULL)
```

Arguments

`post_beta`	the `Nsims x n x K` array of `Nsims` draws from the posterior distribution of the `n x K` matrix of factors, `beta`
`subj`	`n x 1` vector of subject IDs or labels

plot_fitted — *Plot the Bayesian curve fitted values*

Description

Plot the curve posterior means with posterior credible intervals (pointwise and joint), the observed data, and true curves (if known)

Usage

```
plot_fitted(y, mu, postY, y_true = NULL, t01 = NULL,
  include_joint_bands = FALSE)
```

Arguments

`y`	the `n x 1` vector of time series observations
`mu`	the `n x 1` vector of fitted values, i.e., posterior expectation of the mean
`postY`	the `nsims x n` matrix of posterior draws from which to compute intervals
`y_true`	the `n x 1` vector of points along the true curve
`t01`	the observation points; if NULL, assume `n` equally spaced points from 0 to 1
`include_joint_bands`	logical; if TRUE, compute simultaneous credible bands

Examples

```
# FIXME
```

plot_flc — *Plot the factor loading curves*

Description

Plot posterior mean of the factor loading curves together with the simultaneous and pointwise 95% credible bands.

Usage

```
plot_flc(post_fk, tau = NULL)
```

Arguments

`post_fk`	the `Nsims x m x K` array of `Nsims` draws from the posterior distribution of the `m x K` matrix of FLCs, `fk`
`tau`	`m x 1` vector of observation points

`simulate_fosr` *Simulate a function-on-scalar regression model*

Description

Simulate data from a function-on-scalar regression model, allowing for subject-specific random effects. The predictors are multivariate normal with mean zero and covariance `corr^abs(j1-j2)` for correlation parameter `corr` between predictors `j1` and `j2`. More predictors than observations (p > n) is allowed.

Usage

```
simulate_fosr(n = 100, m = 50, RSNR = 5, K_true = 4, p_0 = 1000,
  p_1 = 5, sparse_factors = TRUE, corr = 0, perc_missing = 0)
```

Arguments

`n`	number of observed curves (i.e., number of subjects)
`m`	total number of observation points (i.e., points along the curve)
`RSNR`	root signal-to-noise ratio
`K_true`	rank of the model (i.e., number of basis functions used for the functional data simulations)
`p_0`	number of true zero regression coefficients
`p_1`	number of true nonzero regression coefficients
`sparse_factors`	logical; if TRUE, then for each nonzero predictor j, sample a subset of k=1:K_true factors to be nonzero#'
`corr`	correlation parameter for predictors
`perc_missing`	percentage of missing data (between 0 and 1); default is zero

Value

a list containing the following:

- `Y`: the simulated `n x m` functional data matrix
- `X`: the simulated `n x p` design matrix
- `tau`: the `m`-dimensional vector of observation points
- `Y_true`: the true `n x m` functional data matrix (w/o noise)
- `alpha_tilde_true` the true `m x p` matrix of regression coefficient functions
- `alpha_arr_true` the true `K_true x p` matrix of regression coefficient factors
- `Beta_true` the true `n x K_true` matrix of factors
- `F_true` the true `m x K_true` matrix of basis (loading curve) functions
- `sigma_true` the true observation error standard deviation

Note

The basis functions (or loading curves) are orthonormalized polynomials, so large values of `K_true` are not recommended.

Examples

```
# Example: simulate FOSR
sim_data = simulate_fosr(n = 100, m = 20, p_0 = 100, p_1 = 5)
Y = sim_data$Y; X = sim_data$X; tau = sim_data$tau
```

Index