

Package ‘spBPS’

July 23, 2025

Title Bayesian Predictive Stacking for Scalable Geospatial Transfer Learning

Version 0.0-4

Maintainer Luca Presicce <l.presicce@campus.unimib.it>

Author Luca Presicce [aut, cre] (ORCID:
<<https://orcid.org/0009-0005-7062-3523>>),
Sudipto Banerjee [aut]

Description Provides functions for Bayesian Predictive Stacking within the Bayesian transfer learning framework for geospatial artificial systems, as introduced in “Bayesian Transfer Learning for Artificially Intelligent Geospatial Systems: A Predictive Stacking Approach” (Presicce and Banerjee, 2024) <[doi:10.48550/arXiv.2410.09504](https://doi.org/10.48550/arXiv.2410.09504)>. This methodology enables efficient Bayesian geostatistical modeling, utilizing predictive stacking to improve inference across spatial datasets. The core functions leverage ‘C++’ for high-performance computation, making the framework well-suited for large-scale spatial data analysis in parallel and distributed computing environments. Designed for scalability, it allows seamless application in computationally demanding scenarios.

Depends R (>= 1.8.0)

Imports Rcpp, CVXR, mniw

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, rmarkdown, mvnfast, foreach, parallel, doParallel,
tictoc, MBA, RColorBrewer, classInt, sp, fields, testthat (>= 3.0.0)

Config/testthat/edition 3

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.1

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-10-25 09:20:01 UTC

Contents

arma_dist	3
bayesMvLMconjugate	3
BPS_combine	4
BPS_post	6
BPS_postdraws	7
BPS_postdraws_MvT	8
BPS_post_MvT	9
BPS_pred	10
BPS_pred_MvT	12
BPS_PseudoBMA	14
BPS_weights	15
BPS_weights_MvT	16
conv_opt	17
CVXR_opt	18
dens_kcv	19
dens_kcv_MvT	19
dens_loocv	20
dens_loocv_MvT	20
d_pred_cpp	21
d_pred_cpp_MvT	21
expand_grid_cpp	22
fit_cpp	23
fit_cpp_MvT	23
forceSymmetry_cpp	24
models_dens	24
models_dens_MvT	25
post_draws	25
post_draws_MvT	26
pred_bayesMvLMconjugate	26
r_pred_cond	27
r_pred_cond_MvT	28
r_pred_joint	29
r_pred_joint_MvT	29
r_pred_marg	30
r_pred_marg_MvT	31
sample_index	31
spPredict_BPS	32
subset_data	32

Index

34

arma_dist	<i>Compute the Euclidean distance matrix</i>
-----------	--

Description

Compute the Euclidean distance matrix

Usage

```
arma_dist(X)
```

Arguments

X [matrix](#) (typically of N coordinates on $\mathbb{R}^2$)

Value

[matrix](#) distance matrix of the elements of X

Examples

```
## Compute the Distance matrix of dimension (n x n)
n <- 100
p <- 2
X <- matrix(runif(n*p), nrow = n, ncol = p)
distance.matrix <- arma_dist(X)
```

bayesMvLMconjugate	<i>Gibbs sampler for Conjugate Bayesian Multivariate Linear Models</i>
--------------------	--

Description

Gibbs sampler for Conjugate Bayesian Multivariate Linear Models

Usage

```
bayesMvLMconjugate(Y, X, mu_B, V_B, nu, Psi, n_iter = 1000, burn_in = 500)
```

Arguments

Y	matrix $n \times q$ of response variables
X	matrix $n \times p$ of predictors
mu_B	matrix $p \times q$ prior mean for β
V_B	matrix $p \times p$ prior row covariance for β
nu	double prior parameter for Σ
Psi	matrix prior parameter for Σ
n_iter	integer iteration number for Gibbs sampler
burn_in	integer number of burn-in iteration

Value

B_samples **array** of posterior sample for β
 Sigma_samples **array** of posterior samples for Σ

Examples

```
## Generate data
n <- 100
p <- 3
q <- 2
Y <- matrix(rnorm(n*q), nrow = n, ncol = q)
X <- matrix(rnorm(n*p), nrow = n, ncol = p)

## Prior parameters
mu_B <- matrix(0, p, q)
V_B <- diag(10, p)
nu <- 3
Psi <- diag(q)

## Samples from posteriors
n_iter <- 1000
burn_in <- 500
set.seed(1234)
samples <- spBPS::bayesMvLMconjugate(Y, X, mu_B, V_B, nu, Psi, n_iter, burn_in)
```

BPS_combine

Combine subset models wiht BPS

Description

Combine subset models wiht BPS

Usage

```
BPS_combine(fit_list, K, rp)
```

Arguments

`fit_list` **list** K fitted model outputs composed by two elements each: first named *epd*, second named *W*

`K` **integer** number of folds

`rp` **double** percentage of observations to take into account for optimization (default `t=1`)

Value

matrix posterior predictive density evaluations (each columns represent a different model)

Examples

```
## Generate subsets of data
n <- 100
p <- 3
X <- matrix(rnorm(n*p), nrow = n, ncol = p)
Y <- matrix(rnorm(n), nrow = n, ncol = 1)
crd <- matrix(runif(n*2), nrow = n, ncol = 2)
data_part <- subset_data(data = list(Y = Y, X = X, crd = crd), K = 10)

## Select competitive set of values for hyperparameters
delta_seq <- c(0.1, 0.2, 0.3)
phi_seq <- c(3, 4, 5)

## Perform Bayesian Predictive Stacking within subsets
fit_list <- vector(length = 10, mode = "list")
for (i in 1:10) {
  Yi <- data_part$Y_list[[i]]
  Xi <- data_part$X_list[[i]]
  crd_i <- data_part$crd_list[[i]]
  p <- ncol(Xi)
  bps <- spBPS::BPS_weights(data = list(Y = Yi, X = Xi),
                           priors = list(mu_b = matrix(rep(0, p)),
                                         V_b = diag(10, p),
                                         a = 2,
                                         b = 2), coords = crd_i,
                           hyperpar = list(delta = delta_seq,
                                           phi = phi_seq),
                           K = 5)

  w_hat <- bps$W
  epd <- bps$epd
  fit_list[[i]] <- list(epd, w_hat) }

## Combination weights between partitions using Bayesian Predictive Stacking
comb_bps <- BPS_combine(fit_list = fit_list, K = 10, rp = 1)
```

BPS_post	<i>Perform the BPS sampling from posterior and posterior predictive given a set of stacking weights</i>
----------	---

Description

Perform the BPS sampling from posterior and posterior predictive given a set of stacking weights

Usage

```
BPS_post(data, X_u, priors, coords, crd_u, hyperpar, W, R)
```

Arguments

data	list two elements: first named Y , second named X
X_u	matrix unobserved instances covariate matrix
priors	list priors: named μ_b, V_b, a, b
coords	matrix sample coordinates for X and Y
crd_u	matrix unobserved instances coordinates
hyperpar	list two elements: first named δ , second named ϕ
W	matrix set of stacking weights
R	integer number of desired samples

Value

list BPS posterior predictive samples

Examples

```
## Generate subsets of data
n <- 100
p <- 3
X <- matrix(rnorm(n*p), nrow = n, ncol = p)
Y <- matrix(rnorm(n), nrow = n, ncol = 1)
crd <- matrix(runif(n*2), nrow = n, ncol = 2)
data_part <- subset_data(data = list(Y = Y, X = X, crd = crd), K = 10)

## Select competitive set of values for hyperparameters
delta_seq <- c(0.1, 0.2, 0.3)
phi_seq <- c(3, 4, 5)
## Fit local models
fit_list <- vector(length = 10, mode = "list")
for (i in 1:10) {
  Yi <- data_part$Y_list[[i]]
  Xi <- data_part$X_list[[i]]
  crd_i <- data_part$crd_list[[i]]
  p <- ncol(Xi)
```

```

bps <- spBPS::BPS_weights(data = list(Y = Yi, X = Xi),
                          priors = list(mu_b = matrix(rep(0, p)),
                                         V_b = diag(10, p),
                                         a = 2,
                                         b = 2), coords = crd_i,
                          hyperpar = list(delta = delta_seq,
                                           phi = phi_seq),
                          K = 5)

w_hat <- bps$W
epd <- bps$epd
fit_list[[i]] <- list(epd, w_hat) }

## Model combination weights between partitions using Bayesian Predictive Stacking
comb_bps <- BPS_combine(fit_list = fit_list, K = 10, rp = 1)
Wbps <- comb_bps$W
W_list <- comb_bps$W_list

## Generate prediction points
m <- 100
X_new <- matrix(rnorm(m*p), nrow = m, ncol = p)
crd_new <- matrix(runif(m*2), nrow = m, ncol = 2)

## Perform posterior and posterior predictive sampling
R <- 250
subset_ind <- sample(1:10, R, TRUE, Wbps)
postsmp_and_pred <- vector(length = R, mode = "list")
for (r in 1:R) {
  ind_s <- subset_ind[r]
  Ys <- matrix(data_part$Y_list[[ind_s]])
  Xs <- data_part$X_list[[ind_s]]
  crds <- data_part$crd_list[[ind_s]]
  Ws <- W_list[[ind_s]]
  result <- spBPS::BPS_post(data = list(Y = Ys, X = Xs), coords = crds,
                           X_u = X_new, crd_u = crd_new,
                           priors = list(mu_b = matrix(rep(0, p)),
                                          V_b = diag(10, p),
                                          a = 2,
                                          b = 2),
                           hyperpar = list(delta = delta_seq,
                                           phi = phi_seq),
                           W = Ws, R = 1)

  postsmp_and_pred[[r]] <- result}

```

BPS_postdraws

Compute the BPS posterior samples given a set of stacking weights

Description

Compute the BPS posterior samples given a set of stacking weights

Usage

```
BPS_postdraws(data, priors, coords, hyperpar, W, R)
```

Arguments

data **list** two elements: first named Y , second named X
 priors **list** priors: named μ_b, V_b, a, b
 coords **matrix** sample coordinates for X and Y
 hyperpar **list** two elements: first named δ , second named ϕ
 W **matrix** set of stacking weights
 R **integer** number of desired samples

Value

matrix BPS posterior samples

BPS_postdraws_MvT	<i>Compute the BPS posterior samples given a set of stacking weights</i>
-------------------	--

Description

Compute the BPS posterior samples given a set of stacking weights

Usage

```
BPS_postdraws_MvT(data, priors, coords, hyperpar, W, R, par)
```

Arguments

data **list** two elements: first named Y , second named X
 priors **list** priors: named μ_B, V_r, Ψ, ν
 coords **matrix** sample coordinates for X and Y
 hyperpar **list** two elements: first named α , second named ϕ
 W **matrix** set of stacking weights
 R **integer** number of desired samples
 par if TRUE only β and Σ are sampled (ω is omitted)

Value

matrix BPS posterior samples

BPS_post_MvT	<i>Perform the BPS sampling from posterior and posterior predictive given a set of stacking weights</i>
--------------	---

Description

Perform the BPS sampling from posterior and posterior predictive given a set of stacking weights

Usage

```
BPS_post_MvT(data, X_u, priors, coords, crd_u, hyperpar, W, R)
```

Arguments

data	list two elements: first named Y , second named X
X_u	matrix unobserved instances covariate matrix
priors	list priors: named μ_B, V_r, Ψ, ν
coords	matrix sample coordinates for X and Y
crd_u	matrix unobserved instances coordinates
hyperpar	list two elements: first named α , second named ϕ
W	matrix set of stacking weights
R	integer number of desired samples

Value

list BPS posterior predictive samples

Examples

```
## Generate subsets of data
n <- 100
p <- 3
q <- 2
X <- matrix(rnorm(n*p), nrow = n, ncol = p)
Y <- matrix(rnorm(n*q), nrow = n, ncol = q)
crd <- matrix(runif(n*2), nrow = n, ncol = 2)
data_part <- subset_data(data = list(Y = Y, X = X, crd = crd), K = 10)

## Select competitive set of values for hyperparameters
alfa_seq <- c(0.7, 0.8, 0.9)
phi_seq <- c(3, 4, 5)

## Fit local models
fit_list <- vector(length = 10, mode = "list")
for (i in 1:10) {
  Yi <- data_part$Y_list[[i]]
  Xi <- data_part$X_list[[i]]
```

```

crd_i <- data_part$crd_list[[i]]
bps <- spBPS::BPS_weights_MvT(data = list(Y = Yi, X = Xi),
                             priors = list(mu_B = matrix(0, nrow = p, ncol = q),
                                           V_r = diag(10, p),
                                           Psi = diag(1, q),
                                           nu = 3), coords = crd_i,
                             hyperpar = list(alpha = alfa_seq,
                                              phi = phi_seq),
                             K = 5)

w_hat <- bps$W
epd <- bps$epd
fit_list[[i]] <- list(epd, w_hat) }

## Model combination weights between partitions using Bayesian Predictive Stacking
comb_bps <- BPS_combine(fit_list = fit_list, K = 10, rp = 1)
Wbps <- comb_bps$W
W_list <- comb_bps$W_list

## Generate prediction points
m <- 100
X_new <- matrix(rnorm(m*p), nrow = m, ncol = p)
crd_new <- matrix(runif(m*2), nrow = m, ncol = 2)

## Perform posterior and posterior predictive sampling
R <- 250
subset_ind <- sample(1:10, R, TRUE, Wbps)
postsmpl_and_pred <- vector(length = R, mode = "list")
for (r in 1:R) {
  ind_s <- subset_ind[r]
  Ys <- data_part$Y_list[[ind_s]]
  Xs <- data_part$X_list[[ind_s]]
  crds <- data_part$crd_list[[ind_s]]
  Ws <- W_list[[ind_s]]
  result <- spBPS::BPS_post_MvT(data = list(Y = Ys, X = Xs), coords = crds,
                                X_u = X_new, crd_u = crd_new,
                                priors = list(mu_B = matrix(0, nrow = p, ncol = q),
                                              V_r = diag(10, p),
                                              Psi = diag(1, q),
                                              nu = 3),
                                hyperpar = list(alpha = alfa_seq,
                                              phi = phi_seq),
                                W = Ws, R = 1)

  postsmpl_and_pred[[r]] <- result}

```



```

                                phi = phi_seq),
                                K = 5)

w_hat <- bps$W
epd <- bps$epd
fit_list[[i]] <- list(epd, w_hat) }

## Model combination weights between partitions using Bayesian Predictive Stacking
comb_bps <- BPS_combine(fit_list = fit_list, K = 10, rp = 1)
Wbps <- comb_bps$W
W_list <- comb_bps$W_list

## Generate prediction points
m <- 50
X_new <- matrix(rnorm(m*p), nrow = m, ncol = p)
crd_new <- matrix(runif(m*2), nrow = m, ncol = 2)

## Perform posterior predictive sampling
R <- 250
subset_ind <- sample(1:10, R, TRUE, Wbps)
predictions <- vector(length = R, mode = "list")
for (r in 1:R) {
  ind_s <- subset_ind[r]
  Ys <- matrix(data_part$Y_list[[ind_s]])
  Xs <- data_part$X_list[[ind_s]]
  crds <- data_part$crd_list[[ind_s]]
  Ws <- W_list[[ind_s]]
  result <- spBPS::BPS_pred(data = list(Y = Ys, X = Xs), coords = crds,
                            X_u = X_new, crd_u = crd_new,
                            priors = list(mu_b = matrix(rep(0, p)),
                                           V_b = diag(10, p),
                                           a = 2,
                                           b = 2),
                            hyperpar = list(delta = delta_seq,
                                             phi = phi_seq),
                            W = Ws, R = 1)

  predictions[[r]] <- result}

```

BPS_pred_MvT

Compute the BPS spatial prediction given a set of stacking weights

Description

Compute the BPS spatial prediction given a set of stacking weights

Usage

BPS_pred_MvT(data, X_u, priors, coords, crd_u, hyperpar, W, R)

Arguments

data	list two elements: first named Y , second named X
X_u	matrix unobserved instances covariate matrix
priors	list priors: named μ_B, V_r, Ψ, ν
coords	matrix sample coordinates for X and Y
crd_u	matrix unobserved instances coordinates
hyperpar	list two elements: first named α , second named ϕ
W	matrix set of stacking weights
R	integer number of desired samples

Value

list BPS posterior predictive samples

Examples

```
## Generate subsets of data
n <- 100
p <- 3
q <- 2
X <- matrix(rnorm(n*p), nrow = n, ncol = p)
Y <- matrix(rnorm(n*q), nrow = n, ncol = q)
crd <- matrix(runif(n*2), nrow = n, ncol = 2)
data_part <- subset_data(data = list(Y = Y, X = X, crd = crd), K = 10)

## Select competitive set of values for hyperparameters
alfa_seq <- c(0.7, 0.8, 0.9)
phi_seq <- c(3, 4, 5)

## Fit local models
fit_list <- vector(length = 10, mode = "list")
for (i in 1:10) {
  Yi <- data_part$Y_list[[i]]
  Xi <- data_part$X_list[[i]]
  crd_i <- data_part$crd_list[[i]]
  bps <- spBPS::BPS_weights_MvT(data = list(Y = Yi, X = Xi),
                                priors = list(mu_B = matrix(0, nrow = p, ncol = q),
                                              V_r = diag(10, p),
                                              Psi = diag(1, q),
                                              nu = 3), coords = crd_i,
                                hyperpar = list(alpha = alfa_seq,
                                                phi = phi_seq),
                                K = 5)

  w_hat <- bps$W
  epd <- bps$epd
  fit_list[[i]] <- list(epd, w_hat) }

## Model combination weights between partitions using Bayesian Predictive Stacking
comb_bps <- BPS_combine(fit_list = fit_list, K = 10, rp = 1)
```

```

Wbps <- comb_bps$W
W_list <- comb_bps$W_list

## Generate prediction points
m <- 100
X_new <- matrix(rnorm(m*p), nrow = m, ncol = p)
crd_new <- matrix(runif(m*2), nrow = m, ncol = 2)

## Perform posterior predictive sampling
R <- 250
subset_ind <- sample(1:10, R, TRUE, Wbps)
predictions <- vector(length = R, mode = "list")
for (r in 1:R) {
  ind_s <- subset_ind[r]
  Ys <- data_part$Y_list[[ind_s]]
  Xs <- data_part$X_list[[ind_s]]
  crds <- data_part$crd_list[[ind_s]]
  Ws <- W_list[[ind_s]]
  result <- spBPS::BPS_pred_MvT(data = list(Y = Ys, X = Xs), coords = crds,
                                X_u = X_new, crd_u = crd_new,
                                priors = list(mu_B = matrix(0, nrow = p, ncol = q),
                                              V_r = diag(10, p),
                                              Psi = diag(1, q),
                                              nu = 3),
                                hyperpar = list(alpha = alfa_seq,
                                                phi = phi_seq),
                                W = Ws, R = 1)

  predictions[[r]] <- result}

```

BPS_PseudoBMA

Combine subset models wiht Pseudo-BMA

Description

Combine subset models wiht Pseudo-BMA

Usage

```
BPS_PseudoBMA(fit_list)
```

Arguments

`fit_list` [list](#) K fitted model outputs composed by two elements each: first named *epd*, second named *W*

Value

[matrix](#) posterior predictive density evaluations (each columns represent a different model)

Examples

```
## Generate subsets of data
n <- 100
p <- 3
X <- matrix(rnorm(n*p), nrow = n, ncol = p)
Y <- matrix(rnorm(n), nrow = n, ncol = 1)
crd <- matrix(runif(n*2), nrow = n, ncol = 2)
data_part <- subset_data(data = list(Y = Y, X = X, crd = crd), K = 10)

## Select competitive set of values for hyperparameters
delta_seq <- c(0.1, 0.2, 0.3)
phi_seq <- c(3, 4, 5)

## Perform Bayesian Predictive Stacking within subsets
fit_list <- vector(length = 10, mode = "list")
for (i in 1:10) {
  Yi <- data_part$Y_list[[i]]
  Xi <- data_part$X_list[[i]]
  crd_i <- data_part$crd_list[[i]]
  p <- ncol(Xi)
  bps <- spBPS::BPS_weights(data = list(Y = Yi, X = Xi),
                           priors = list(mu_b = matrix(rep(0, p)),
                                         V_b = diag(10, p),
                                         a = 2,
                                         b = 2), coords = crd_i,
                           hyperpar = list(delta = delta_seq,
                                           phi = phi_seq),
                           K = 5)

  w_hat <- bps$W
  epd <- bps$epd
  fit_list[[i]] <- list(epd, w_hat) }

## Combination weights between partitions using Pseudo Bayesian Model Averaging
comb_bps <- BPS_PseudoBMA(fit_list = fit_list)
```

BPS_weights

Compute the BPS weights by convex optimization

Description

Compute the BPS weights by convex optimization

Usage

```
BPS_weights(data, priors, coords, hyperpar, K)
```

Arguments

data **list** two elements: first named Y , second named X
 priors **list** priors: named μ_b, V_b, a, b
 coords **matrix** sample coordinates for X and Y
 hyperpar **list** two elements: first named δ , second named ϕ
 K **integer** number of folds

Value

matrix posterior predictive density evaluations (each columns represent a different model)

Examples

```
## Generate subsets of data
n <- 100
p <- 3
X <- matrix(rnorm(n*p), nrow = n, ncol = p)
Y <- matrix(rnorm(n), nrow = n)
crd <- matrix(runif(n*2), nrow = n, ncol = 2)

## Select competitive set of values for hyperparameters
delta_seq <- c(0.1, 0.2, 0.3)
phi_seq <- c(3, 4, 5)

## Perform Bayesian Predictive Stacking within subsets
bps <- spBPS::BPS_weights(data = list(Y = Y, X = X),
                           priors = list(mu_b = matrix(rep(0, p)),
                                           V_b = diag(10, p),
                                           a = 2,
                                           b = 2), coords = crd,
                           hyperpar = list(delta = delta_seq,
                                           phi = phi_seq),
                           K = 5)
```

BPS_weights_MvT

Compute the BPS weights by convex optimization

Description

Compute the BPS weights by convex optimization

Usage

```
BPS_weights_MvT(data, priors, coords, hyperpar, K)
```

Arguments

data **list** two elements: first named Y , second named X
 priors **list** priors: named μ_B, V_r, Ψ, ν
 coords **matrix** sample coordinates for X and Y
 hyperpar **list** two elements: first named α , second named ϕ
 K **integer** number of folds

Value

matrix posterior predictive density evaluations (each columns represent a different model)

Examples

```
## Generate subsets of data
n <- 100
p <- 3
q <- 2
X <- matrix(rnorm(n*p), nrow = n, ncol = p)
Y <- matrix(rnorm(n*q), nrow = n, ncol = q)
crd <- matrix(runif(n*2), nrow = n, ncol = 2)

## Select competitive set of values for hyperparameters
alfa_seq <- c(0.7, 0.8, 0.9)
phi_seq <- c(3, 4, 5)

## Perform Bayesian Predictive Stacking within subsets
bps <- spBPS::BPS_weights_MvT(data = list(Y = Y, X = X),
                              priors = list(mu_B = matrix(0, nrow = p, ncol = q),
                                             V_r = diag(10, p),
                                             Psi = diag(1, q),
                                             nu = 3), coords = crd,
                              hyperpar = list(alpha = alfa_seq,
                                              phi = phi_seq),
                              K = 5)
```

conv_opt

Solver for Bayesian Predictive Stacking of Predictive densities convex optimization problem

Description

Solver for Bayesian Predictive Stacking of Predictive densities convex optimization problem

Usage

```
conv_opt(scores)
```

Arguments

scores **matrix** $N \times K$ of expected predictive density evaluations for the K models considered

Value

W **matrix** of Bayesian Predictive Stacking weights for the K models considered

Examples

```
## Generate (randomly) K predictive scores for n observations
n <- 50
K <- 5
scores <- matrix(runif(n*K), nrow = n, ncol = K)

## Find Bayesian Predictive Stacking weights
opt_weights <- conv_opt(scores)
```

CVXR_opt

Compute the BPS weights by convex optimization

Description

Compute the BPS weights by convex optimization

Usage

```
CVXR_opt(scores)
```

Arguments

scores **matrix** $N \times K$ of expected predictive density evaluations for the K models considered

Value

conv_opt **function** to perform convex optimization with CVXR R package

dens_kcv	<i>Compute the KCV of the density evaluations for fixed values of the hyperparameters</i>
----------	---

Description

Compute the KCV of the density evaluations for fixed values of the hyperparameters

Usage

```
dens_kcv(data, priors, coords, hyperpar, K)
```

Arguments

data	list two elements: first named Y , second named X
priors	list priors: named μ_b, V_b, a, b
coords	matrix sample coordinates for X and Y
hyperpar	list two elemets: first named δ , second named ϕ
K	integer number of folds

Value

[vector](#) posterior predictive density evaluations

dens_kcv_MvT	<i>Compute the KCV of the density evaluations for fixed values of the hyperparameters</i>
--------------	---

Description

Compute the KCV of the density evaluations for fixed values of the hyperparameters

Usage

```
dens_kcv_MvT(data, priors, coords, hyperpar, K)
```

Arguments

data	list two elements: first named Y , second named X
priors	list priors: named μ_B, V_r, Ψ, ν
coords	matrix sample coordinates for X and Y
hyperpar	list two elemets: first named α , second named ϕ
K	integer number of folds

Value

[vector](#) posterior predictive density evaluations

dens_loocv	<i>Compute the LOOCV of the density evaluations for fixed values of the hyperparameters</i>
------------	---

Description

Compute the LOOCV of the density evaluations for fixed values of the hyperparameters

Usage

```
dens_loocv(data, priors, coords, hyperpar)
```

Arguments

data	list two elements: first named Y , second named X
priors	list priors: named μ_b, V_b, a, b
coords	matrix sample coordinates for X and Y
hyperpar	list two elemets: first named δ , second named ϕ

Value

[vector](#) posterior predictive density evaluations

dens_loocv_MvT	<i>Compute the LOOCV of the density evaluations for fixed values of the hyperparameters</i>
----------------	---

Description

Compute the LOOCV of the density evaluations for fixed values of the hyperparameters

Usage

```
dens_loocv_MvT(data, priors, coords, hyperpar)
```

Arguments

data	list two elements: first named Y , second named X
priors	list priors: named μ_B, V_r, Ψ, ν
coords	matrix sample coordinates for X and Y
hyperpar	list two elemets: first named α , second named ϕ

Value

[vector](#) posterior predictive density evaluations

d_pred_cpp	<i>Evaluate the density of a set of unobserved response with respect to the conditional posterior predictive</i>
------------	--

Description

Evaluate the density of a set of unobserved response with respect to the conditional posterior predictive

Usage

```
d_pred_cpp(data, X_u, Y_u, d_u, d_us, hyperpar, poster)
```

Arguments

data	list two elements: first named Y , second named X
X_u	matrix unobserved instances covariate matrix
Y_u	matrix unobserved instances response matrix
d_u	matrix unobserved instances distance matrix
d_us	matrix cross-distance between unobserved and observed instances matrix
hyperpar	list two elements: first named δ , second named ϕ
poster	list output from <code>fit_cpp</code> function

Value

[vector](#) posterior predictive density evaluations

d_pred_cpp_MvT	<i>Evaluate the density of a set of unobserved response with respect to the conditional posterior predictive</i>
----------------	--

Description

Evaluate the density of a set of unobserved response with respect to the conditional posterior predictive

Usage

```
d_pred_cpp_MvT(data, X_u, Y_u, d_u, d_us, hyperpar, poster)
```

Arguments

data	list two elements: first named Y , second named X
X_u	matrix unobserved instances covariate matrix
Y_u	matrix unobserved instances response matrix
d_u	matrix unobserved instances distance matrix
d_us	matrix cross-distance between unobserved and observed instances matrix
hyperpar	list two elements: first named α , second named ϕ
poster	list output from <code>fit_cpp</code> function

Value

[double](#) posterior predictive density evaluation

expand_grid_cpp	<i>Build a grid from two vector (i.e. equivalent to <code>expand.grid()</code> in R)</i>
-----------------	--

Description

Build a grid from two vector (i.e. equivalent to `expand.grid()` in R)

Usage

```
expand_grid_cpp(x, y)
```

Arguments

x	vector first vector of numeric elements
y	vector second vector of numeric elements

Value

[matrix](#) expanded grid of combinations

Examples

```
## Create a matrix from all combination of vectors
x <- seq(0, 10, length.out = 100)
y <- seq(-1, 1, length.out = 20)
grid <- expand_grid_cpp(x = x, y = y)
```

fit_cpp	<i>Compute the parameters for the posteriors distribution of β and Σ (i.e. updated parameters)</i>
---------	---

Description

Compute the parameters for the posteriors distribution of β and Σ (i.e. updated parameters)

Usage

```
fit_cpp(data, priors, coords, hyperpar)
```

Arguments

data	list two elements: first named Y , second named X
priors	list priors: named μ_b, V_b, a, b
coords	matrix sample coordinates for X and Y
hyperpar	list two elemets: first named δ , second named ϕ

Value

[list](#) posterior update parameters

fit_cpp_MvT	<i>Compute the parameters for the posteriors distribution of β and Σ (i.e. updated parameters)</i>
-------------	---

Description

Compute the parameters for the posteriors distribution of β and Σ (i.e. updated parameters)

Usage

```
fit_cpp_MvT(data, priors, coords, hyperpar)
```

Arguments

data	list two elements: first named Y , second named X
priors	list priors: named μ_B, V_r, Ψ, ν
coords	matrix sample coordinates for X and Y
hyperpar	list two elemets: first named α , second named ϕ

Value

[list](#) posterior update parameters

forceSymmetry_cpp	<i>Function to subset data for meta-analysis</i>
-------------------	--

Description

Function to subset data for meta-analysis

Usage

```
forceSymmetry_cpp(mat)
```

Arguments

mat **matrix** not-symmetric matrix

Value

matrix symmetric matrix (lower triangular of mat is used)

Examples

```
## Force matrix to be symmetric (avoiding numerical problems)
n <- 4
X <- matrix(runif(n*n), nrow = n, ncol = n)
X <- forceSymmetry_cpp(mat = X)
```

models_dens	<i>Return the CV predictive density evaluations for all the model combinations</i>
-------------	--

Description

Return the CV predictive density evaluations for all the model combinations

Usage

```
models_dens(data, priors, coords, hyperpar, useKCV, K)
```

Arguments

data	list two elements: first named Y , second named X
priors	list priors: named μ_b, V_b, a, b
coords	matrix sample coordinates for X and Y
hyperpar	list two elemets: first named δ , second named ϕ
useKCV	if TRUE K-fold cross validation is used instead of LOOCV (no default)
K	integer number of folds

Value

[matrix](#) posterior predictive density evaluations (each columns represent a different model)

models_dens_MvT	<i>Return the CV predictive density evaluations for all the model combinations</i>
-----------------	--

Description

Return the CV predictive density evaluations for all the model combinations

Usage

```
models_dens_MvT(data, priors, coords, hyperpar, useKCV, K)
```

Arguments

data [list](#) two elements: first named Y , second named X
 priors [list](#) priors: named μ_B, V_r, Ψ, ν
 coords [matrix](#) sample coordinates for X and Y
 hyperpar [list](#) two elemets: first named α , second named ϕ
 useKCV if TRUE K-fold cross validation is used instead of LOOCV (no default)
 K [integer](#) number of folds

Value

[matrix](#) posterior predictive density evaluations (each columns represent a different model)

post_draws	<i>Sample R draws from the posterior distributions</i>
------------	--

Description

Sample R draws from the posterior distributions

Usage

```
post_draws(poster, R, par, p)
```

Arguments

poster [list](#) output from `fit_cpp` function
 R [integer](#) number of posterior samples
 par if TRUE only β and σ^2 are sampled (ω is omitted)
 p [integer](#) if par = TRUE, it specifies the column number of X

Value

[list](#) posterior samples

post_draws_MvT	<i>Sample R draws from the posterior distributions</i>
----------------	--

Description

Sample R draws from the posterior distributions

Usage

```
post_draws_MvT(poster, R, par, p)
```

Arguments

poster	list output from fit_cpp function
R	integer number of posterior samples
par	if TRUE only β and Σ are sampled (ω is omitted)
p	integer if par = TRUE, it specifies the column number of X

Value

[list](#) posterior samples

pred_bayesMvLMconjugate	<i>Predictive sampler for Conjugate Bayesian Multivariate Linear Models</i>
-------------------------	---

Description

Predictive sampler for Conjugate Bayesian Multivariate Linear Models

Usage

```
pred_bayesMvLMconjugate(X_new, B_samples, Sigma_samples)
```

Arguments

X_new	matrix $n_{new} \times p$ of predictors for new data points
B_samples	array of posterior sample for β
Sigma_samples	array of posterior samples for Σ

Value

Y_pred **matrix** of posterior mean for response matrix Y predictions

Y_pred_samples **array** of posterior predictive sample for response matrix Y

Examples

```
## Generate data
n <- 100
p <- 3
q <- 2
Y <- matrix(rnorm(n*q), nrow = n, ncol = q)
X <- matrix(rnorm(n*p), nrow = n, ncol = p)

## Prior parameters
mu_B <- matrix(0, p, q)
V_B <- diag(10, p)
nu <- 3
Psi <- diag(q)

## Samples from posteriors
n_iter <- 1000
burn_in <- 500
set.seed(1234)
samples <- spBPS::bayesMvLMconjugate(Y, X, mu_B, V_B, nu, Psi, n_iter, burn_in)

## Extract posterior samples
B_samples <- samples$B_samples
Sigma_samples <- samples$Sigma_samples

## Samples from predictive posterior (based posterior samples)
m <- 50
X_new <- matrix(rnorm(m*p), nrow = m, ncol = p)
pred <- spBPS::pred_bayesMvLMconjugate(X_new, B_samples, Sigma_samples)
```

r_pred_cond	<i>Draw from the conditional posterior predictive for a set of unobserved covariates</i>
-------------	--

Description

Draw from the conditional posterior predictive for a set of unobserved covariates

Usage

```
r_pred_cond(data, X_u, d_u, d_us, hyperpar, poster, post)
```

Arguments

<code>data</code>	list two elements: first named Y , second named X
<code>X_u</code>	matrix unobserved instances covariate matrix
<code>d_u</code>	matrix unobserved instances distance matrix
<code>d_us</code>	matrix cross-distance between unobserved and observed instances matrix
<code>hyperpar</code>	list two elemets: first named δ , second named ϕ
<code>poster</code>	list output from <code>fit_cpp</code> function
<code>post</code>	list output from <code>post_draws</code> function

Value

[list](#) posterior predictive samples

<code>r_pred_cond_MvT</code>	<i>Draw from the conditional posterior predictive for a set of unobserved covariates</i>
------------------------------	--

Description

Draw from the conditional posterior predictive for a set of unobserved covariates

Usage

```
r_pred_cond_MvT(data, X_u, d_u, d_us, hyperpar, poster, post)
```

Arguments

<code>data</code>	list two elements: first named Y , second named X
<code>X_u</code>	matrix unobserved instances covariate matrix
<code>d_u</code>	matrix unobserved instances distance matrix
<code>d_us</code>	matrix cross-distance between unobserved and observed instances matrix
<code>hyperpar</code>	list two elemets: first named α , second named ϕ
<code>poster</code>	list output from <code>fit_cpp_MvT</code> function
<code>post</code>	list output from <code>post_draws_MvT</code> function

Value

[list](#) posterior predictive samples

r_pred_joint	<i>Draw from the joint posterior predictive for a set of unobserved covariates</i>
--------------	--

Description

Draw from the joint posterior predictive for a set of unobserved covariates

Usage

```
r_pred_joint(data, X_u, d_u, d_us, hyperpar, poster, R)
```

Arguments

data	list two elements: first named Y , second named X
X_u	matrix unobserved instances covariate matrix
d_u	matrix unobserved instances distance matrix
d_us	matrix cross-distance between unobserved and observed instances matrix
hyperpar	list two elemets: first named δ , second named ϕ
poster	list output from <code>fit_cpp</code> function
R	integer number of posterior predictive samples

Value

[list](#) posterior predictive samples

r_pred_joint_MvT	<i>Draw from the joint posterior predictive for a set of unobserved covariates</i>
------------------	--

Description

Draw from the joint posterior predictive for a set of unobserved covariates

Usage

```
r_pred_joint_MvT(data, X_u, d_u, d_us, hyperpar, poster, R)
```

Arguments

<code>data</code>	list two elements: first named Y , second named X
<code>X_u</code>	matrix unobserved instances covariate matrix
<code>d_u</code>	matrix unobserved instances distance matrix
<code>d_us</code>	matrix cross-distance between unobserved and observed instances matrix
<code>hyperpar</code>	list two elemets: first named α , second named ϕ
<code>poster</code>	list output from <code>fit_cpp</code> function
<code>R</code>	integer number of posterior predictive samples

Value

[list](#) posterior predictive samples

<code>r_pred_marg</code>	<i>Draw from the marginals posterior predictive for a set of unobserved covariates</i>
--------------------------	--

Description

Draw from the marginals posterior predictive for a set of unobserved covariates

Usage

```
r_pred_marg(data, X_u, d_u, d_us, hyperpar, poster, R)
```

Arguments

<code>data</code>	list two elements: first named Y , second named X
<code>X_u</code>	matrix unobserved instances covariate matrix
<code>d_u</code>	matrix unobserved instances distance matrix
<code>d_us</code>	matrix cross-distance between unobserved and observed instances matrix
<code>hyperpar</code>	list two elemets: first named δ , second named ϕ
<code>poster</code>	list output from <code>fit_cpp</code> function
<code>R</code>	integer number of posterior predictive samples

Value

[list](#) posterior predictive samples

r_pred_marg_MvT	<i>Draw from the joint posterior predictive for a set of unobserved covariates</i>
-----------------	--

Description

Draw from the joint posterior predictive for a set of unobserved covariates

Usage

```
r_pred_marg_MvT(data, X_u, d_u, d_us, hyperpar, poster, R)
```

Arguments

data	list two elements: first named Y , second named X
X_u	matrix unobserved instances covariate matrix
d_u	matrix unobserved instances distance matrix
d_us	matrix cross-distance between unobserved and observed instances matrix
hyperpar	list two elemets: first named α , second named ϕ
poster	list output from <code>fit_cpp</code> function
R	integer number of posterior predictive samples

Value

[list](#) posterior predictive samples

sample_index	<i>Function to sample integers (index)</i>
--------------	--

Description

Function to sample integers (index)

Usage

```
sample_index(size, length, p)
```

Arguments

size	integer dimension of the set to sample
length	integer number of elements to sample
p	vector sampling probabilities

Value

[vector](#) sample of integers

spPredict_BPS	<i>Perform prediction for BPS accelerated models - loop over prediction set</i>
---------------	---

Description

Perform prediction for BPS accelerated models - loop over prediction set

Usage

```
spPredict_BPS(data, X_u, priors, coords, crd_u, hyperpar, W, R, J)
```

Arguments

data	list two elements: first named Y , second named X
X_u	matrix unobserved instances covariate matrix
priors	list priors: named μ_b, V_b, a, b
coords	matrix sample coordinates for X and Y
crd_u	matrix unobserved instances coordinates
hyperpar	list two elements: first named δ , second named ϕ
W	matrix set of stacking weights
R	integer number of desired samples
J	integer number of desired partition of prediction set

Value

[list](#) BPS posterior predictive samples

subset_data	<i>Function to subset data for meta-analysis</i>
-------------	--

Description

Function to subset data for meta-analysis

Usage

```
subset_data(data, K)
```

Arguments

data	list three elements: first named Y , second named X , third named crd
K	integer number of desired subsets

Value

[list](#) subsets of data, and the set of indexes

Examples

```
## Create a list of K random subsets given a list with Y, X, and crd
n <- 100
p <- 3
q <- 2
X <- matrix(rnorm(n*p), nrow = n, ncol = p)
Y <- matrix(rnorm(n*q), nrow = n, ncol = q)
crd <- matrix(runif(n*2), nrow = n, ncol = 2)
subsets <- subset_data(data = list(Y = Y, X = X, crd = crd), K = 10)
```

Index

arma_dist, 3
array, 4, 26, 27

bayesMvLMconjugate, 3
BPS_combine, 4
BPS_post, 6
BPS_post_MvT, 9
BPS_postdraws, 7
BPS_postdraws_MvT, 8
BPS_pred, 10
BPS_pred_MvT, 12
BPS_PseudoBMA, 14
BPS_weights, 15
BPS_weights_MvT, 16

conv_opt, 17
CVXR_opt, 18

d_pred_cpp, 21
d_pred_cpp_MvT, 21
dens_kcv, 19
dens_kcv_MvT, 19
dens_loocv, 20
dens_loocv_MvT, 20
double, 4, 5, 22

expand_grid_cpp, 22

fit_cpp, 23
fit_cpp_MvT, 23
forceSymmetry_cpp, 24
function, 18

integer, 4–6, 8, 9, 11, 13, 16, 17, 19, 24–26, 29–32

list, 5, 6, 8, 9, 11, 13, 14, 16, 17, 19–26, 28–33

matrix, 3–6, 8, 9, 11, 13, 15–32
models_dens, 24

models_dens_MvT, 25

post_draws, 25
post_draws_MvT, 26
pred_bayesMvLMconjugate, 26

r_pred_cond, 27
r_pred_cond_MvT, 28
r_pred_joint, 29
r_pred_joint_MvT, 29
r_pred_marg, 30
r_pred_marg_MvT, 31

sample_index, 31
spPredict_BPS, 32
subset_data, 32

vector, 19–22, 31