Package ‘BayesSUR’

April 17, 2020

Type Package
Title Bayesian Seemingly Unrelated Regression
Version 1.1-1
Date 2020-04-17
Description Bayesian seemingly unrelated regression with general variable selection and dense/sparse covariance matrix. The sparse seemingly unrelated regression is described in Banterle et al. (2018) <doi:10.1101/467019>.
License MIT + file LICENSE
Copyright The C++ files pugixml.cpp, pugixml.hpp and pugiconfig.hpp are Copyright (C) 2006-2018 by Arseny Kapoulkine (arseny.kapoulkine@gmail.com) and Copyright (C) 2003 by Kristen Wegner (kristen@tima.net). The R function vertical.image.legend() has Copyright (C) 2013 by Jenise Swall (jswall@vcu.edu).
VignetteBuilder R.rsp
RoxygenNote 7.1.0
Depends R (>= 3.6.0)
Encoding UTF-8
LinkingTo Rcpp, RcppArmadillo (>= 0.9.000)
Imports Rcpp, xml2, igraph, Matrix, tikzDevice, stats, utils, grDevices, graphics
Suggests R.rsp
LazyData true
NeedsCompilation yes
SystemRequirements C++11
Author Marco Banterle [aut], Zhi Zhao [aut, cre], Leonardo Bottolo [ctb], Sylvia Richardson [ctb], Alex Lewin [aut], Manuela Zucknick [ctb]
BayesSUR

Maintainer Zhi Zhao <zhi.zhao@medisin.uio.no>
Repository CRAN
Date/Publication 2020-04-17 18:50:02 UTC

R topics documented:

BayesSUR ................................................. 2
BayesSUR_internal ..................................... 6
coef.BayesSUR ........................................... 7
eLPD ....................................................... 8
example_eQTL ........................................... 9
example_GDSC ........................................... 14
fitted.BayesSUR ......................................... 18
getEstimator ............................................. 19
plot.BayesSUR ........................................... 20
plot.CPO .................................................. 21
plot.Estimator .......................................... 23
plot.Manhattan .......................................... 25
plot.MCMCdiag .......................................... 26
plot.Network ............................................ 27
plot.ResponseGraph .................................... 30
predict.BayesSUR ....................................... 31
print.BayesSUR ......................................... 32
summary.BayesSUR ..................................... 33
targetGene ............................................... 34

Index 39

BayesSUR  main function of the package

Description

Main function of the package. Fits a range of models introduced in the package vignette BayesSUR.pdf. Returns an object of S3 class BayesSUR. There are three options for the prior on the residual covariance matrix (i.e., independent inverse-Gamma, inverse-Wishart and hyper-inverse Wishart) and three options for the prior on the latent indicator variable (i.e., independent Bernoulli, hotspot and Markov random field). So there are nine models in total. See details for their combinations.

Usage

BayesSUR(
  Y,
  X,
  X_0 = NULL,
  data = NULL,
BayesSUR

outFilePath = "",
nIter = 10000,
burnin = 5000,
nChains = 2,
covariancePrior = "HIW",
gammaPrior = "",
betaPrior = "independent",
gammaSampler = "bandit",
gammaInit = "MLE",
mrfG = NULL,
standardize = TRUE,
standardize.response = TRUE,
maxThreads = 2,
output.gamma = TRUE,
output.beta = TRUE,
output.G = TRUE,
output_sigmaRho = TRUE,
output.pi = TRUE,
output_tail = TRUE,
output_model_size = TRUE,
output_model_visit = FALSE,
output_CPO = FALSE,
output_Y = TRUE,
output_X = TRUE,
hyperpar = list(),
tmpFolder = "tmp/"
)

Arguments

Y, X, X_0  vectors of indexes (with respect to the data matrix) for the outcomes, the covariates to select and the fixed covariates respectively if data is either a path to a file or a matrix; if the 'data' argument is not provided, these needs to be matrices containing the data instead.
data  a data frame if using formula. If not using formula, it is either a matrix/dataframe or the path to (a plain text) data file with variables on the columns and observations on the rows.
outFilePath  path to where the output files are to be written. The default path is the current working directory.
nIter  number of iterations for the MCMC procedure.
burnin  number of iterations (or fraction of iterations) to discard at the start of the chain. Default is 0.
nChains  number of parallel chains to run.
covariancePrior  string indicating the prior for the covariance $\Sigma$; it has to be either "HIW" for the hyper-inverse-Wishar (which will result in a sparse covariance matrix), "IW" for the inverse-Wishart prior (dense covariance) or "IG" for independent
inverse-Gamma on all the diagonal elements and 0 otherwise. See the details for the model specification

**gammaPrior** string indicating the gamma prior to use, either "hotspot" for the Hotspot prior of Bottolo (2011), "MRF" for the Markov Random Field prior or "hierarchical" for a simpler hierarchical prior. See the details for the model specification

**betaPrior** string indicating the beta prior to use, either "independent" for the independent spike-and-slab prior or "reGroup" for the random effects for $X_0$ and independent spike-and-slab priors for other predictors

**gammaSampler** string indicating the type of sampler for gamma, either "bandit" for the Thompson sampling inspired sampler or "MC3" for the usual $MC^3$ sampler

**gammaInit** gamma initialisation to either all-zeros ("0"), all ones ("1"), randomly ("R") or (default) MLE-informed ("MLE").

**mrfG** either a matrix or a path to the file containing the G matrix for the MRF prior on gamma (if necessary)

**standardize** logical flag for X variable standardization. Default is standardize=TRUE. The coefficients are returned on the standardized scale.

**standardize.response** Standardization for the response variables. Default is standardize.response=TRUE.

**maxThreads** maximum threads used for parallelization. Default is 2.

**output_gamma** allow ( TRUE ) or suppress ( FALSE ) the output for gamma. See the return value below for more information.

**output_beta** allow ( TRUE ) or suppress ( FALSE ) the output for beta. See the return value below for more information.

**output_G** allow ( TRUE ) or suppress ( FALSE ) the output for G. See the return value below for more information.

**output_sigmaRho** allow ( TRUE ) or suppress ( FALSE ) the output for sigmaRho. See the return value below for more information.

**output_pi** allow ( TRUE ) or suppress ( FALSE ) the output for pi. See the return value below for more information.

**output_tail** allow ( TRUE ) or suppress ( FALSE ) the output for tail (hotspot tail probability). See the return value below for more information.

**output_model_size** allow ( TRUE ) or suppress ( FALSE ) the output for model_size. See the return value below for more information.

**output_model_visit** allow ( TRUE ) or suppress ( FALSE ) the output for all visited models over the MCMC iterations. Default is FALSE. See the return value below for more information.

**output_CPO** allow ( TRUE ) or suppress ( FALSE ) the output for *; possible outputs are gamma, G, beta, sigmaRho, pi, tail (hotspot tail probability), model_size, CPO. See the return value below for more information.

**output_Y** allow ( TRUE ) or suppress ( FALSE ) the output for responses dataset Y.

**output_X** allow ( TRUE ) or suppress ( FALSE ) the output for predictors dataset X.
hyperpar  a list of named hyperparameters to use instead of the default values. Valid names are mrf_d, mrf_e, a_sigma, b_sigma, a_tau, b_tau, nu, a_eta, b_eta, a_o, b_o, a_pi, b_pi, a_w and b_w. Their default values are a_w=2, b_w=5, a_omega=1, b_omega=1, a_o=2, b_o=p-2, a_pi=2, b_pi=1, nu=s+2, a_tau=0.1, b_tau=10, a_eta=0.1, b_eta=1, a_sigma=1, b_sigma=1, mrf_d=-3 and mrf_e=0.03. See the vignette for more information.

tmpFolder  the path to a temporary folder where intermediate data files are stored (will be erased at the end of the chain) default to local tmpFolder

Details

The arguments covariancePrior and gammaPrior specify the model HRR, dSUR or SSUR with different gamma prior. Let $\gamma_{jk}$ be latent indicator variable of each coefficient and $C$ be covariance matrix of response variables. The nine models specified through the arguments covariancePrior and gammaPrior are as follows.

<table>
<thead>
<tr>
<th>$\gamma_{jk}$-Bernoulli</th>
<th>$\gamma_{jk}$-hotspot</th>
<th>$\gamma_{jk}$-MRF</th>
</tr>
</thead>
<tbody>
<tr>
<td>C~indep</td>
<td>HRR-B</td>
<td>HRR-H</td>
</tr>
<tr>
<td>C~IW</td>
<td>dSUR-B</td>
<td>dSUR-H</td>
</tr>
<tr>
<td>C~HIW</td>
<td>SSUR-B</td>
<td>SSUR-H</td>
</tr>
</tbody>
</table>

Value

An object of class "BayesSUR":

- status - the running status
- input - a list of all input parameters by the user
- output - a list of the all output filenames:
  - "*_logP_out.txt" - contains each row for the 1000t-th iteration's log-likelihoods of parameters, i.e., Tau, Eta, JunctionTree, SigmaRho, O, Pi, Gamma, W, Beta and data conditional log-likelihood depending on the models.
  - "*_gamma_out.txt" - posterior mean of the latent indicator matrix.
  - "*_pi_out.txt" - posterior mean of the predictor effects (propensity) by decomposing the probability of the latent indicator.
  - "*_hotspot_tail_p_out.txt" - posterior mean of the hotspot tail probability. Only available for the hotspot prior on the gamma.
  - "*_beta_out.txt" - posterior mean of the coefficients matrix.
  - "*_G_out.txt" - posterior mean of the response graph. Only available for the HIW prior on the covariance.
  - "*_sigmaRho_out.txt" - posterior mean of the transformed parameters. Not available for the IG prior on the covariance.
  - "*_model_size_out.txt" - contains each row for the 1000t-th iteration's model sizes of the multiple response variables.
  - "*_model_visit_g_out.txt" - contains each row for the nonzero indices of the vectorized estimated graph matrix for each iteration.
  - "*_model_visit_gamma_out.txt" - contains each row for the nonzero indices of the vectorized estimated gamma matrix for each iteration.
- "*_CPO_out.txt" - the (scaled) conditional predictive ordinates (CPO).
- "*_CPOsumy_out.txt" - the (scaled) conditional predictive ordinates (CPO) with joint posterior predictive of the response variables.
- "*_WAIC_out.txt" - the widely applicable information criterion (WAIC).
- "*_Y.txt" - responses dataset.
- "*_X.txt" - predictors dataset.
- "*_X0.txt" - fixed predictors dataset.

- call - the matched call.

References

Examples
```r
data("example_eQTL", package = "BayesSUR")
hyperpar <- list(a_w = 2, b_w = 5)
set.seed(9173)
fit <- BayesSUR(Y = example_eQTL["blockList"][[1]],
X = example_eQTL["blockList"][[2]],
data = example_eQTL["data"], outFilePath = tempdir(),
nIter = 100, burnin = 50, nChains = 2, gammaPrior = "hotspot",
hyperpar = hyperpar, tmpFolder = "tmp/", output_CPO=TRUE)

## check output
# show the summary information
summary(fit)

# show the estimated beta, gamma and graph of responses Gy
estimators <- getEstimator(fit, estimator=c("beta","gamma","Gy"))
plot.Estimator(estimators)

plot.Estimator(estimators, fig.tex = TRUE)
system(paste(getOption("pdfviewer"), "FaramEstimator.pdf"))
```
**Description**

Run a SUR Bayesian sampler – internal function

**Arguments**

- `dataFile` path to data file
- `outFilePath` path to where the output is to be written
- `nIter` number of iterations
- `nChains` number of parallel chains to run

**coef.BayesSUR**

`coef.BayesSUR` extract the posterior mean of the coefficients of a "BayesSUR" class object

**Description**

Extract the posterior mean of the coefficients of a "BayesSUR" class object

**Usage**

```r
## S3 method for class 'BayesSUR'
coef(object, Pmax = 0, ...)
```

**Arguments**

- `object` an object of class "BayesSUR"
- `Pmax` threshold that truncates the estimated coefficients based on thresholding the estimated latent indicator variable. Default is 0.
- `...` other arguments

**Value**

Estimated coefficients are from an object of class "BayesSUR". If the BayesSUR specified data standardization, the fitted values are base based on standardized data.

**Examples**

```r
data("example_eQTL", package = "BayesSUR")
hyperpar <- list( a_w = 2 , b_w = 5 )

set.seed(9173)
fit <- BayesSUR(Y = example_eQTL[["blockList"]][[1]],
                X = example_eQTL[["blockList"]][[2]],
                data = example_eQTL[["data"]], outFilePath = tempdir(),
                nIter = 100, burnin = 50, nChains = 2, gammaPrior = "hotspot",
                hyperpar = hyperpar, tmpFolder = "tmp/"
```
elpd measure the prediction accuracy by the expected log pointwise predictive density

Description

Measure the prediction accuracy by the elpd (expected log pointwise predictive density). The out-of-sample predictive fit can either be estimated by Bayesian leave-one-out cross-validation (LOO) or by widely applicable information criterion (WAIC) (Vehtari et al. 2017).

Usage

elpd(object, method = "LOO")

Arguments

object an object of class "BayesSUR"
method the name of the prediction accuracy index. Default is the "LOO" (Bayesian LOO estimate of out-of-sample predictive fit). The other index is the "WAIC" (widely applicable information criterion). For the HRR models, both "LOO" and "WAIC" are computed based on the multivariate t-distribution of the posterior predictive rather than approximation of importance sampling.

Value

Return the prediction accuracy measure from an object of class "BayesSUR". It is elpd.loo if the argument method="LOO" and elpd.WAIC if method="WAIC".

References


Examples

data("example_eQTL", package = "BayesSUR")
hyperpar = list( a_w = 2 , b_w = 5 )
set.seed(9173)
fit <- BayesSUR(Y = example_eQTL[['blockList']][[1]],
               X = example_eQTL[['blockList']][[2]],
data = example_eQTL[['data']], outFilePath = tempdir(),
nIter = 100, burnin = 50, nChains = 2, gammaPrior = "hotspot",
hyperpar = hyperpar, tmpFolder = "tmp/", output_CPO=TRUE)
### Description

Simulated data set to mimic a small expression quantitative trait loci (eQTL) example, with $p=150$ single nucleotide polymorphisms (SNPs) as explanatory variables, $s=10$ gene expression features as response variables and data for $n=100$ observations. Loading the data will load the associated blockList object needed to fit the model with BayesSUR(). The R code for generating the simulated data is given in the Examples paragraph.

```r
#importFrom BDgraph rgwish #importFrom gRbase mcsMAT #importFrom scrime simulateSNPs
```

### Usage

```r
eexample_eQTL
```

### Format

An object of class list of length 4.

### Examples

```r
# Load the eQTL sample dataset
data("example_eQTL", package = "BayesSUR")
str(example_eQTL)

## Not run:
#===============
# The code below is to show how to generate the dataset "example_eQTL.rda" above
#===============

requireNamespace("BDgraph", quietly = TRUE)
requireNamespace("gRbase", quietly = TRUE)
requireNamespace("scrime", quietly = TRUE)

#---------------------
# Problem Dimensions
n = 100
p = 150
s = 10
#---------------------

# Select a set of $n \times p$ (SNPs) covariates
## The synthetic data in the paper use a subset of the real SNPs as covariates, 
but as the NFBC66 dataset is confidential we'll use scrime to sample similar data

```r
x = scrime::simulateSNPs(c(n,10), p, c(3, 2), prop.explain = c(0.9, 0.95))$data[1:n,]
x = cbind(rep(1,n),x)
```

```r
Graph_pattern = 2  # in 2,3,4
snr = 25  # in 5,15,25
corr_param = 0.9  # in 0.3 , 0.6 , 0.9
```

### Create the underlying graph
```r
if(graph_pattern==1){
  ### 1) Random but full
  G = matrix(1,s,s)
  Prime = list(c(1:s))
  Res = Prime
  Sep = list()
}
```
```r
else if(graph_pattern==2){
  ### 2) Block Diagonal structure
  Prime = list(c(1:floor(s*2/3)),
               c((floor(s*2/3)+1):(ceiling(s*4/5)-1)),
               c(ceiling(s*4/5):s))
  Res = Prime
  Sep = lapply(Res,function(x) which(x==-99))
  G = matrix(0,s,s)
  for(i in Prime){
    G[i,i] = 1
  }
}
```
```r
else if(graph_pattern==3){
  ### 3) Decomposable model
  Prime = list(c(1:floor(s*5/12),ceiling(s*9/10):s),
               c(floor(s*2/9):(ceiling(s*2/3)-1)),
               c(ceiling(s*2/3):(ceiling(s*4/5)-1)),
               c(ceiling(s*4/5):s))
  Sep = list(); H=list()
  for( i in 2:length(Prime)){
    H = union(H,Prime[[i-1]])
    Sep[[i-1]] = intersect( H,Prime[[i]])
  }
  Res = list()
```
```r
Res[[1]] = Prime[[1]]
for( i in 2:length(Prime)){
  Res[[i]] = setdiff( Prime[[i]], Sep[[i-1]])
}

G = matrix(0,s,s)
for(i in Prime)
  G[i,i] = 1

## decomp check
if(graph_pattern==4){
  nblocks = 5
  nElemPerBlock = c(floor(s/4),floor(s/2)-1-floor(s/4),
                    ceiling(s*2/3)-1-floor(s/2),7)
  nElemPerBlock = c(nElemPerBlock, s-sum(nElemPerBlock))
  blockIdx = list()
  for(i in 1:nblocks){
    blockIdx[[i]] = sample(res,nElemPerBlock[i])
  }
  G = matrix(0,s,s)
  ## add diagonal
  for(i in 1:nblocks)
    G[blockIdx[[i]],blockIdx[[i]]] = 1
  ## add cycle
  G[blockIdx[[1]],blockIdx[[2]]] = 1 ; G[blockIdx[[2]],blockIdx[[1]]] = 1
  G[blockIdx[[1]],blockIdx[[5]]] = 1 ; G[blockIdx[[5]],blockIdx[[1]]] = 1
  G[blockIdx[[2]],blockIdx[[3]]] = 1 ; G[blockIdx[[3]],blockIdx[[2]]] = 1
  G[blockIdx[[3]],blockIdx[[5]]] = 1 ; G[blockIdx[[5]],blockIdx[[3]]] = 1
}

## Gamma Pattern
gamma = matrix(0,p+1,s)
gamma[1,] = 1
```

### 2) Extra Patterns
## outcomes (correlated in the decompos model) have some predictors in common
\[
\gamma[6:10, 6:9] = 1
\]

## outcomes (correlated in the decompos model) have some predictors in common
\[
\gamma[16:20, 14:15] = 1
\]

## outcomes (sort-of correlated [pair-wise] in the decompos model)
# have predictors in common 6:15
\[
\gamma[26:30, 4:8] = 1
\]

## outcomes (NOT correlated in the decompos model) have predictors in common 16:17
\[
\gamma[36:40, c(3:5, 9:10)] = 1
\]

## these predictors are associated with ALL the outcomes
\[
\gamma[46:50,] = 1
\]

\[
\text{combn}11 = \text{combn}(\text{rep}(6:9-1)*p, \text{each}=\text{length}(6:10-1)) + \text{rep}(6:10-1, \text{times}=\text{length}(6:9)), 2)
\]
\[
\text{combn}31 = \text{combn}(\text{rep}(4:8-1)*p, \text{each}=\text{length}(26:30-1)) + \text{rep}(26:30-1, \text{times}=\text{length}(4:8)), 2)
\]
\[
\text{combn}32 = \text{combn}(\text{rep}(4:8-1)*p, \text{each}=\text{length}(46:50-1)) + \text{rep}(46:50-1, \text{times}=\text{length}(4:8)), 2)
\]
\[
\text{combn}41 = \text{combn}(\text{rep}(3:5-1)*p, \text{each}=\text{length}(36:40-1)) + \text{rep}(36:40-1, \text{times}=\text{length}(3:5)), 2)
\]
\[
\text{combn}42 = \text{combn}(\text{rep}(3:5-1)*p, \text{each}=\text{length}(46:50-1)) + \text{rep}(46:50-1, \text{times}=\text{length}(3:5)), 2)
\]
\[
\text{combn}51 = \text{combn}(\text{rep}(9:10-1)*p, \text{each}=\text{length}(36:40-1)) + \text{rep}(36:40-1, \text{times}=\text{length}(9:10)), 2)
\]
\[
\text{combn}52 = \text{combn}(\text{rep}(9:10-1)*p, \text{each}=\text{length}(46:50-1)) + \text{rep}(46:50-1, \text{times}=\text{length}(9:10)), 2)
\]

\[
\text{Gmrf} = \text{rbind}(t(\text{combn}11), t(\text{combn}31), t(\text{combn}32), t(\text{combn}41), t(\text{combn}42), t(\text{combn}51), t(\text{combn}52))
\]

## get for every correlated bunch in the decomposable model,
if(graph_pattern<4){
    # a different set of predictors
    for(i in 1:length(Prime))
        gamma[6:10 + (i+6) * 10, Prime[i]] = 1  ## for each Prime component

    ## for every Residual instead
    for(i in 1:length(Res))
        gamma[6:10 + (i+10) * 10, Res[i]] = 1
}
else{

    for(i in 1:length(Prime))
        gamma[6:10 + (i+4) * 10, Prime[i]] = 1  ## for each Prime component

    ## for every Residual instead
    for(i in 1:length(Res))
        gamma[6:10 + (i+9) * 10, Res[i]] = 1
}

#### Sample the betas
sd_b = 1
b = matrix(rnorm((p+1)*s,0,sd_b),p+1,s)
xb = matrix(NA,n,s)
for(i in 1:s){
    if(sum(gamma[,i])>1){
        xb[,i] = x[,gamma[,i]==1] %*% b[gamma[,i]==1,i]
    }else{
        xb[,i] = rep(1,n) * b[1,i]
    }
}

##Sample the variance
v_r = mean(diag(var(xb))) / snr

nu = s+1

M = matrix(corr_param,s,s)
diag(M) = rep(1,s)

P = BDgraph::rgwish(n=1,adj=G,b=3,D=v_r*M)

var = solve(P)

factor = 10 ; factor_min = 0.01; factor_max = 1000
count = 0 ; maxit = 10000

factor_prev = 1

repeat{
    var = var / factor * factor_prev

    ### Sample the errors and the Ys
    cVar = chol(as.matrix(var))
    #err = matrix(rnorm(n*s),n,s) %*% cVar
    err = matrix(rnorm(n*s,sd=0.5),n,s) %*% cVar
    y = xb+err

    ### Reparametrisation ( assuming PEO is 1:s )
    cVar = t(cVar) # make it lower-tri
    S = diag(diag(cVar))
    sigma = S*S
    L = cVar %*% solve(S)
    rho = diag(s) - solve(L)

    ### S/N Ratio
    emp_snr = mean( diag( var(xb) %*% solve(sigma) ))
    emp_g_snr = mean( diag( var( err)**2 %*% rho ) %*% solve(sigma) ))

    if( abs(emp_snr - snr) < (snr/10) | count > maxit ){
        break
    }else{
        if( emp_snr < snr ){ # increase factor
            factor_min = factor
            factor = factor_max
            count = 0
        }
    }
}
else{ # decrease factor
    factor_max = factor
}
factor_prev = factor
factor = (factor_min + factor_max)/2
}
count = count+1
}

colnames(y) <- paste("GEX",1:ncol(y),sep="")
colnames(G) <- colnames(y); Gy <- G
gamma <- gamma[-1,]
mrfG <- Gmrf[!duplicated(Gmrf),]
data = cbind(y,x[,-1]) # leave out the intercept because is coded inside already

example_eQTL = list(data=data, blockList=list(1:s,s+1:p))

## Write data file to the user's directory by save()

## End(Not run)

---
example_GDSC  
Preprocessed data set to mimic a small pharmacogenetic example

Description

Preprocessed data set to mimic a small pharmacogenetic example from the Genomics of Drug Sensitivity in Cancer (GDSC) database, with p=850 gene features as explanatory variables, s=7 drugs sensitivity data as response variables and data for n=498 cell lines. Gene features include p1=343 gene expression features (GEX), p2=426 by copy number variations (CNV) and p3=68 mutated genes (MUT). Loading the data will load the associated blockList (and mrfG) objects needed to fit the model with BayesSUR(). The R code for generating the simulated data is given in the Examples paragraph.

#importFrom plyr mapvalues #importFrom data.table like

Usage

example_GDSC

Format

An object of class list of length 3.
Examples

# Load the GDSC sample dataset
data("example_GDSC", package = "BayesSUR")
str(example_GDSC)

## Not run:
#===============
# This code below is to do preprocessing of GDSC data and obtain the complete dataset
# "example_GDSC.rda" above. The user needs load the datasets from
# ftp://ftp.sanger.ac.uk/pub4/cancerrxgene/releases/release-5.0/.
# But downloading and transforming the three used datasets below to *.csv files first.
#===============

requireNamespace("plyr", quietly = TRUE)
requireNamespace("data.table", quietly = TRUE)

features <- data.frame(read.csv("gdsc_en_input_w5.csv", head=T))
names.fea <- strsplit(rownames(features), "")
features <- t(features)
p <- c(13321, 13747-13321, 13818-13747)
Cell.Line <- rownames(features)
features <- data.frame(Cell.Line, features)

ic50_00 <- data.frame(read.csv("gdsc_drug_sensitivity_fitted_data_w5.csv", head=T))
ic50_0 <- ic50_00[,c(1,4,7)]
drug.id <- data.frame(read.csv("gdsc_tissue_output_w5.csv", head=T))[,c(1,3)]
drug.id <- drug.id[!duplicated(drug.id$drug.id),]
# delete drug.id=1066 since ID1066 and ID156 both correspond drug AZD6482,
# and no ID1066 in the "suppl.Data1" by Garnett et al. (2012)
drug.id2 <- drug.id2[drug.id2$drug.id!=1066,]
drug.id2$drug.name <- as.character(drug.id2$drug.name)
drug.id2$drug.name <- substr(drug.id2$drug.name, 1, nchar(drug.id2$drug.name)-6)
drug.id2$drug.name <- gsub(" ", ",", drug.id2$drug.name)

ic50 <- ic50_0
# mapping the drug_id to drug names in drug sensitivity data set
ic50$drug_id <- plyr::mapvalues(ic50$drug_id, from = drug.id2[,2], to = drug.id2[,1])
colnames(ic50) <- c("Cell.Line", "compound", "IC50")

# transform drug sensitivity overall cell lines to a data matrix
y0 <- reshape(ic50, v.names="IC50", timevar="compound", idvar="Cell.Line", direction="wide")
y0$Cell.Line <- gsub(" ", ",", y0$Cell.Line)

# select nonmissing pharmacological data
y00 <- y0
m0 <- dim(y0)[2]-1
eps <- 0.05
# r1.na is better to be not smaller than r2.na
r1.na <- 0.3
r2.na <- 0.2
k <- 1
while(sum(is.na(y0[,2:(1+m0)]))>0)
  r1.na <- r1.na - eps/k
  r2.na <- r1.na - eps/k
  k <- k + 1
  ## select drugs with <30% (decreasing with k) missing data overall cell lines
  na.y <- apply(y0[,2:(1+m0)], 2, function(xx) sum(is.na(xx))/length(xx))
  while(sum(na.y<r1.na)<m0){
    y0 <- y0[-(1+which(na.y>=r1.na))]
    m0 <- sum(na.y>=r1.na)
    na.y <- apply(y0[,2:(1+m0)], 2, function(xx) sum(is.na(xx))/length(xx))
  }
  ## select cell lines with treatment of at least 80% (increasing with k) drugs
  na.y0 <- apply(y0[,2:(1+m0)], 1, function(xx) sum(is.na(xx))/length(xx))
  while(sum(na.y0<r2.na)<dim(y0)[1]){y0 <- y0[na.y0<r2.na,]
    na.y0 <- apply(y0[,2:(1+m0)], 1, function(xx) sum(is.na(xx))/length(xx))
  }
  num.na <- sum(is.na(y0[,2:(1+m0)]))
  message("#{NA}=", num.na, "\n", "r1.na =", r1.na, ", r2.na =", r2.na, "\n")
}

# combine drug sensitivity, tissues and molecular features
yx <- merge(y0, features, by="Cell.Line")
names.cell.line <- yx$Cell.Line
names.drug <- colnames(yx)[2:(dim(y0)[2])]
names.drug <- substr(names.drug, 6, nchar(names.drug))
# numbers of gene expression features, copy number features and mutation features
p <- c(13321, 13747-13321, 13818-13747)
num.nonpen <- 13
yx <- data.matrix(yx[,,-1])
y <- yx[,1:(dim(y0)[2]-1)]
x <- cbind(yx[,dim(y0)[2]-1+sum(p)+1:num.nonpen], yx[,dim(y0)[2]-1+1:sum(p)])
# delete genes with only one mutated cell line
GDSC <- list(y=y, x=x, p=p, num.nonpen=num.nonpen, names.cell.line=names.cell.line, names.drug=names.drug)

# select a small set of drugs
name_drugs <- c("Methotrexate","RDEA119","PD-0325901","CI-1040", "AZD6244","Nilotinib", "Axitinib")
# example_GDSC

# extract the drugs' pharmacological profiling and tissue dummy

```
col_filter <- colnames(GDSC$y) %in% paste("IC50", name_drugs, sep="")
YX0 <- cbind(
  GDSC$y[, col_filter][, c(1, 3, 6, 4, 7, 2, 5)],
  GDSC$x[, 1:GDSC$num.nonpen]
)
colnames(YX0) <- c(name_drugs, colnames(GDSC$x)[1:GDSC$num.nonpen])
```

# extract the genetic information of CNV & MUT

```
colnames(X23)[1:p[2]] <- paste(substr(colnames(X23)[1:p[2]], 1, nchar(colnames(X23)[1:p[2]] )-3), ".CNV", sep="")
```

# locate all genes with CNV or MUT information

```
name_genes_duplicate <- c(
  substr(colnames(X23)[1:p[2]], 1, nchar(colnames(X23)[1:p[2]] )-4),
)
name_genes <- name_genes_duplicate[!duplicated(name_genes_duplicate)]
```

# select the GEX which have the common genes with CNV or MUT

```
col_filter <- GDSC$num.nonpen + which(
  colnames(GDSC$x)[GDSC$num.nonpen+1:p[1]] %in% name_genes
)
X1 <- GDSC$x[, col_filter]
p[1] <- ncol(X1)
X1 <- log2(X1)
```

# summary the data information

```
example_GDSC <- list( data=cbind( YX0, X1, X23 ) )
example_GDSC$blockList <- list(1:length(name_drugs),
  length(name_drugs)+1:GDSC$num.nonpen,
  ncol(YX0)+1:sum(p))
```

# construct the G matrix: edge potentials in the MRF prior

# edges between drugs: Group1 ("RDEA119", "17-AAG", "PD-0325901", "CI-1040", "AZD6244")
# indexed as (2:5)

```
pathway_genes <- read.table("MAPK_pathway.txt")[[1]]
X1_names_dup <- c(colnames(X1), name_genes_duplicate)
Idx_Pathway1 <- which(X1_names_dup %in% pathway_genes)
rep1 <- rep(Idx_Pathway1, each=length(2:5))
rep2 <- rep((2:5-1) * sum(p), times=length(Idx_Pathway1))
rep3 <- rep1 + rep2
Gmrf_Group1Pathway1 <- t(combn(rep3, 2))
```

# edges between drugs: Group2 ("Nilotinib", "Axitinib") indexed as (6:7)
# delete gene ABL2

```
Idx_Pathway2 <- which(X1_names_dup %like% "BCR" | X1_names_dup %like% "ABL")[-c(3,5)]
Gmrf_Group2Pathway2 <- t(combn(rep(Idx_Pathway2,each=length(6:7))) +
```

# edges between the common gene in different data sources

```r
gmrf_commonGene <- NULL
list_commonGene <- list(0)
k <- 1
for(i in 1:length(name_genes)){
  Idx_CommonGene <- which(c(colnames(X1),name_genes_duplicate) == name_genes[i])
  if(length(Idx_CommonGene) > 1){
    gmrf_commonGene <- rbind(gmrf_commonGene,t(combn(rep(Idx_CommonGene,each=length(name_drugs))
      + rep((1:length(name_drugs)-1)*sum(p),times=length(Idx_CommonGene)), 2)))
    k <- k+1
  }
}
gmrf_duplicate <- rbind( gmrf_group1Pathway1, gmrf_group2Pathway2, gmrf_commonGene )
gmrf <- gmrf_duplicate[!duplicated(gmrf_duplicate),]
example_GDSC$mrfG <- gmrf
```

# create the target gene names of the two groups of drugs

targetGenes1 <- matrix(Idx_Pathway1,nrow=1)
colnames(targetGenes1) <- colnames(example_GDSC$data)[seq_along(targetGene$group1)]
targetGenes2 <- matrix(Idx_Pathway2,nrow=1)
colnames(targetGenes2) <- colnames(example_GDSC$data)[seq_along(targetGene$group2)]
targetGene <- list(group1=targetGenes1, group2=targetGenes2)

## Write data file example_GDSC.rda to the user's directory by save()

## End(Not run)

---

**fitted.BayesSUR**

*fitted response values corresponds to the posterior mean estimates*

### Description

Return the fitted response values that correspond to the posterior mean estimates from a "BayesSUR" class object.

### Usage

```r
## S3 method for class 'BayesSUR'
fitted(object, Pmax = 0, ...)
```

### Arguments

- **object**: an object of class "BayesSUR"
- **Pmax**: threshold that truncates the estimated coefficients based on thresholding the estimated latent indicator variable. Default is 0.
- **...**: other arguments
getEstimator

Value
Fitted values extracted from an object of class "BayesSUR". If the BayesSUR specified data standardization, the fitted values are base based on standardized data.

Examples

data("example_eQTL", package = "BayesSUR")
hyperpar <- list(a_w = 2, b_w = 5)
set.seed(9173)
fit <- BayesSUR(Y = example_eQTL[['blockList']][[1]],
X = example_eQTL[['blockList']][[2]],
data = example_eQTL[['data']], outFilePath = tempdir(),
nIter = 100, burnin = 50, nChains = 2, gammaPrior = "hotspot",
hyperpar = hyperpar, tmpFolder = "tmp/"
)

## check fitted values
fitted.val <- fitted(fit)

---

gEstimator | extract the posterior mean of the parameters

Description
Extract the posterior mean of the parameters of a "BayesSUR" class object.

Usage
getEstimator(object, estimator = "gamma", Pmax = 0)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>an object of class &quot;BayesSUR&quot;</td>
</tr>
<tr>
<td>estimator</td>
<td>the name of one estimator. Default is the latent indicator estimator &quot;gamma&quot;. Other options &quot;beta&quot;, &quot;Gy&quot;, &quot;CPO&quot; and &quot;logP&quot; correspond the posterior means of coefficient matrix, response graph and conditional predictive ordinate (CPO) respectively</td>
</tr>
<tr>
<td>Pmax</td>
<td>threshold that truncate the estimator. Default is 0. If the estimator is beta, then beta is truncated based on the latent indicator matrix shresholding at Pmax</td>
</tr>
</tbody>
</table>

Value
Return the one estimator from an object of class "BayesSUR". It is the posterior mean of the latent indicator variable if estimator="gamma", posterior mean of the regression coefficients if estimator="beta", posterior mean of the response graph if estimator="Gy" and the CPO if estimator="CPO". 
Examples

data("example_eQTL", package = "BayesSUR")
hyperpar <- list( a_w = 2 , b_w = 5 )
set.seed(9173)
fit <- BayesSUR(Y = example_eQTL["blockList" ][[1]],
X = example_eQTL["blockList" ][[2]],
data = example_eQTL["data" ], outFilePath = tempdir(),
nIter = 100, burnin = 50, nChains = 2, gammaPrior = "hotspot",
hyperpar = hyperpar, tmpFolder = "tmp/"
)

## check output
# extract the posterior mean of the coefficients matrix
beta_hat <- getEstimator(fit, estimator="beta")

Description

Convenience function to create a selection of plots for a "BayesSUR" class object. They are plots of estimators, response graph, network, manhattan and MCMC diagnosis indexed by numbers 1:5.

Usage

## S3 method for class 'BayesSUR'
plot(x, which = c(1L:4L), ...)

Arguments

x an object of class "BayesSUR".
which if a subset of the plots is required, specify a subset of the numbers 1:5 which are plots of estimators, response graph, network, manhattan and MCMC diagnosis, respectively. Default is c(1L:4L) Only c(1, 4, 5) is valid for the HRR models.
...
other arguments

Examples

data("example_eQTL", package = "BayesSUR")
hyperpar <- list( a_w = 2 , b_w = 5 )
set.seed(9173)
fit <- BayesSUR(Y = example_eQTL["blockList" ][[1]],
X = example_eQTL["blockList" ][[2]],
data = example_eQTL["data" ], outFilePath = tempdir(),
nIter = 100, burnin = 50, nChains = 2, gammaPrior = "hotspot",
hyperpar = hyperpar, tmpFolder = "tmp/" )
## check output
# Show the interactive plots. Note that it needs at least 2000*(nbloc+1) iterations
# for the diagnosis plots where nbloc=3 by default

plot.BayesSUR(fit)

---

**plot.CPO**  
*plot the conditional predictive ordinate*

### Description

Plot the conditional predictive ordinate (CPO) for each individual of a fitted model generated by BayesSUR which is a "BayesSUR" object. CPO is a handy posterior predictive check because it may be used to identify outliers, influential observations, and for hypothesis testing across different non-nested models (Gelfand 1996).

### Usage

```r
## S3 method for class 'Var'
plot.

plot(
    x,
    outlier.mark = TRUE,
    outlier.thresh = 0.01,
    scale.CPO = TRUE,
    x.loc = FALSE,
    axis.label = NULL,
    las = 0,
    cex.axis = 1,
    mark.pos = c(0, -0.01),
    mark.color = 2,
    mark.cex = 0.8,
    xlab = "Observations",
    ylab = NULL,
    ...
)
```

### Arguments

- **x**: an object of class `getEstimator` with `estimator="CPO"
- **outlier.mark**: mark the outliers with the response names. The default is `FALSE`
- **outlier.thresh**: threshold for the CPOs. The default is `0.01`
- **scale.CPO**: scaled CPOs which is divided by their maximum. The default is `TRUE`
- **x.loc**: a vector of features distance
axis.label  a vector of predictor names which are shown in CPO plot. The default is NULL only showing the indices. The value "auto" show the predictor names from the orginal data.

las  graphical parameter of plot.default

cex.axis  graphical parameter of plot.default

mark.pos  the location of the marked text relative to the point

mark.color  the color of the marked text. The default color is red.

mark.cex  the fontsize of the marked text. The default fontsize is 0.8.

xlab  a title for the x axis

ylab  a title for the y axis

...  other arguments

Details

The default threshold for the CPOs to detect the outliers is 0.01 by Congdon (2005). It can be tuned by the argument outlier.thresh.

References

Statisticat, LLC (2013). *Bayesian Inference*. Farmington, CT: Statisticat, LLC.


Examples

data("example_eQTL", package = "BayesSUR")
hyperpar <- list(a_w = 2 , b_w = 5 )

set.seed(9173)
fit <- BayesSUR(Y = example_eQTL["blockList"][[1]],
X = example_eQTL["blockList"][[2]],
data = example_eQTL["data"], outFilePath = tempdir(),
nIter = 100, burnin = 50, nChains = 2, gammaPrior = "hotspot",
hyperpar = hyperpar, tmpFolder = "tmp/", output_CPO=TRUE)

## check output
# plot the conditional predictive ordinate (CPO)
CPO <- getEstimator(fit, estimator="CPO")
plot(CPO)
plot.Estimator

plot the posterior mean estimators

Description

Plot the posterior mean estimators from a "BayesSUR" class object, including the coefficients beta, latent indicator variable gamma and graph of responses.

Usage

```r
## S3 method for class 'Estimator'
plot(
  x,
  estimator = NULL,
  colorScale.gamma = grey((100:0)/100),
  colorScale.beta = c("blue", "white", "red"),
  legend.cex.axis = 1,
  name.responses = NA,
  name.predictors = NA,
  xlab = "",
  ylab = "",
  fig.tex = FALSE,
  output = "ParamEstimator",
  header = "",
  header.cex = 2,
  tick = FALSE,
  mgp = c(2.5, 1, 0),
  title.beta = paste("Estimator", "$\hat{\bm{B}}$"),
  title.gamma = paste("Estimator", "$\hat{\mathbf{\Gamma}}$"),
  title.Gy = paste("Estimator", "$\hat{\mathcal{G}}$"),
  cex.main = 1.5,
  ...
)
```

Arguments

- `x`: an object of class `getEstimator` with `estimator=c("beta","gamma","Gy")`
- `estimator`: print the heatmap of estimators. The value "beta" is for the estimated coefficients matrix, "gamma" for the latent indicator matrix and "Gy" for the graph of responses
- `colorScale.gamma`: value palette for gamma
- `colorScale.beta`: a vector of three colors for diverging color schemes
- `legend.cex.axis`: magnification of axis annotation relative to cex
name.responses  a vector of the response names. The default is "NA" only to show the locations. The value "auto" show the response names from the orginal data.

name.predictors  a vector of the predictor names. The default is "NA" only to show the locations. The value "auto" show the predictor names from the orginal data.

xlab  a title for the x axis

ylab  a title for the y axis

fig.tex  print the figure through LaTex. Default is "FALSE"

output  the file name of printed figure

header  the main title

header.cex  size of the main title for all estimators

tick  a logical value specifying whether tickmarks and an axis line should be drawn. Default is "FALSE"

mgp  the margin line (in mex units) for the axis title, axis labels and axis line

title.beta  a title for the printed "beta" if fig.tex=TRUE

title.gamma  a title for the printed "gamma" if fig.tex=TRUE

title.Gy  a title for the printed "Gy" if fig.tex=TRUE

cex.main  size of the title for each estimator

...  other arguments

Examples

data("example_eQTL", package = "BayesSUR")
hyperpar <- list( a_w = 2 , b_w = 5 )

set.seed(9173)
fit <- BayesSUR(Y = example_eQTL[["blockList"]][[1]],
X = example_eQTL[["blockList"]][[2]],
data = example_eQTL[["data"]], outFilePath = tempdir(),
nIter = 100, burnin = 50, nChains = 2, gammaPrior = "hotspot",
hyperpar = hyperpar, tmpFolder = "tmp/"
)

## check output
# Plot the estimators from the fitted object

estimators <- getEstimator(fit, estimator = c("beta","gamma","Gy"))
plot(estimators)

plot(estimators, fig.tex = TRUE)
system(paste(getOption("pdfviewer"), "ParamEstimator.pdf"))
plot.Manhattan  

plot Manhattan-like plots for marginal posterior inclusion probabilities (mPIP) and numbers of responses of association for predictors

Description

Plot Manhattan-like plots for marginal posterior inclusion probabilities (mPIP) and numbers of responses of association for predictors of a "BayesSUR" class object.

Usage

```r
## S3 method for class 'Manhattan'
plot(
x, which = c(1, 2),
x.loc = FALSE,
axis.label = "auto",
mark.responses = NULL,
xlab1 = "Predictors",
ylab1 = "mPIP",
xlab2 = "Predictors",
ylab2 = "No. of responses",
threshold = 0.5,
las = 0,
cex.axis = 1,
mark.pos = c(0, 0),
mark.color = 2,
mark.cex = 0.8,
header = "",
...)
```

Arguments

- `x`: an object of class `getEstimator` with `estimator="gamma"
- `which`: if it's value "1" showing the Manhattan-like plot of the marginal posterior inclusion probabilities (mPIP). If it's value "2" showing the Manhattan-like plot of the number of responses. The default is to show both figures.
- `x.loc`: a vector of features distance
- `axis.label`: a vector of predictor names which are shown in the Manhattan-like plot. The value "NULL" only showing the indices. The default "auto" show the predictor names from the orginal data.
- `mark.responses`: a vector of response names which are shown in the Manhattan-like plot for the mPIP
- `xlab1`: a title for the x axis of Manhattan-like plot for the mPIP
- `ylab1`: a title for the y axis of Manhattan-like plot for the mPIP
- `ylab2`: a title for the y axis of Manhattan-like plot for the number of responses
### plot.MCMCdiag

Show trace plots and diagnostic density plots of a fitted model object of class "BayesSUR".

#### Examples

```r
data("example_eQTL", package = "BayesSUR")
hyperpar <- list( a_w = 2 , b_w = 5 )

set.seed(9173)
fit <- BayesSUR(Y = example_eQTL[["blockList"]][[1]],
                X = example_eQTL[["blockList"]][[2]],
                data = example_eQTL[["data"]], outFilePath = tempdir(),
                nIter = 100, burnin = 50, nChains = 2, gammaPrior = "hotspot",
                hyperpar = hyperpar, tmpFolder = "tmp/" )

## check output
# show the Manhattan-like plots
gamma <- getEstimator(fit, estimator="gamma")
plot(gamma)
```

---

#### Description

Show trace plots and diagnostic density plots of a fitted model object of class "BayesSUR".

#### Usage

```r
## S3 method for class 'MCMCdiag'
plot(x, nbloc = 3, HIWg = NULL, header = "", ...)```
plot.Network

Arguments

- `x`: an object of class `getEstimator` with `estimator="logP"`
- `nbloc`: number of splits for the last half iterations after substracting burn-in length
- `HIWg`: diagnostic plot of the response graph. Default is `NULL`. `HIW="degree"` prints the diagnostic of the degrees of response nodes. `HIW="edges"` prints the diagnostic of every edge between two responses. `HIW="lik"` prints the diagnostic of the posterior likelihoods of the hyperparameters related to the response relationships
- `header`: the main title
- `...`: other arguments for the plots of the log-likelihood and model size

Examples

data("example_eQTL", package = "BayesSUR")
hyperpar <- list( a_w = 2, b_w = 5 )
set.seed(9173)
fit <- BayesSUR(Y = example_eQTL["blockList"][[1]],
                X = example_eQTL["blockList"][[2]],
                data = example_eQTL["data"], outFilePath = tempdir(),
                nIter = 100, burnin = 50, nChains = 2, gammaPrior = "hotspot",
                hyperpar = hyperpar, tmpFolder = "tmp/"
)
## check output
MCMCdiag <- getEstimator(fit, estimator = "logP")
plot(MCMCdiag)

plot.Network

plot the network representation of the associations between responses and predictors

Description

Plot the network representation of the associations between responses and predictors, based on the estimated gamma matrix and graph of responses from a "BayesSUR" class object.

Usage

```r
## S3 method for class 'Network'
plot(  
x,  
includeResponse = NULL,  
excludeResponse = NULL,  
includePredictor = NULL,  
excludePredictor = NULL,  
MatrixGamma = NULL,
```


PmaxPredictor = 0.5,
PmaxResponse = 0.5,
nodesizePredictor = 2,
nodesizeResponse = 15,
no.isolates = FALSE,
lineup = 1.2,
gray.alpha = 0.6,
edgewidth.response = 5,
edgewidth.predictor = 2,
edge.weight = FALSE,
label.predictor = NULL,
label.response = NULL,
color.predictor = NULL,
color.response = NULL,
name.predictors = NULL,
name.responses = NULL,
vertex.frame.color = NA,
layoutInCircle = FALSE,
header = "",
... )

Arguments

x an object of class getEstimator with estimator=c("gamma","Gy")
includeResponse
   A vector of the response names which are shown in the network
excludeResponse
   A vector of the response names which are not shown in the network
includePredictor
   A vector of the predictor names which are shown in the network
excludePredictor
   A vector of the predictor names which are not shown in the network
MatrixGamma
   A matrix or dataframe of the latent indicator variable. Default is NULL and to
   extract it from object of class inheriting from an object of class "BayesSUR"
PmaxPredictor
   cutpoint for thresholding the estimated latent indicator variable. Default is 0.5
PmaxResponse
   cutpoint for thresholding the learning structure matrix of multiple response vari-
   ables. Default is 0.5
nodesizePredictor
   node size of Predictors in the output graph. Default is 15
nodesizeResponse
   node size of response variables in the output graph. Default is 25
no.isolates
   remove isolated nodes from responses graph and Full graph, may get problem if
   there are also isolated Predictors
lineup
   A ratio of the heights between responses’ area and Predictors’
gray.alpha
   the opacity. The default is 0.6
plot.Network

edgewith.response
the edge width between response nodes

edgewith.predictor
the edge width between the predictor and response node

draw weighted edges after thresholding at 0.5. The default value FALSE is not to
draw weighted edges

label.predictor
A vector of the names of predictors

label.response
A vector of the names of response variables

color.predictor
color of the predictor nodes

color.response
color of the response nodes

name.predictors
a subtitle for the predictors

name.responses
a subtitle for the responses

vertex.frame.color
The color of the frame of the vertices. If you don’t want vertices to have a frame,
supply NA as the color name

layoutInCircle
place vertices on a circle, in the order of their vertex ids. The default is FALSE

header
the main title

Examples

data("example_eQTL", package = "BayesSUR")
hyperpar <- list( a_w = 2 , b_w = 5 )

set.seed(9173)
fit <- BayesSUR(Y = example_eQTL["blockList"][[1]],
X = example_eQTL["blockList"][[2]],
data = example_eQTL["data"], outFilePath = tempdir(),
nIter = 100, burnin = 50, nChains = 2, gammaPrior = "hotspot",
hyperpar = hyperpar, tmpFolder = "tmp/"
)

# check output
# show the Network representation of the associations between responses and features
network <- getEstimator(fit, estimator = c("gamma","Gy"))
plot(network)
**plot.ResponseGraph**

*plot the estimated graph for multiple response variables*

**Description**

Plot the estimated graph for multiple response variables from a "BayesSUR" class object.

**Usage**

```r
## S3 method for class 'ResponseGraph'
plot(
  x,
  PmaxResponse = 0.5,
  PtrueResponse = NULL,
  name.responses = NA,
  edge.weight = FALSE,
  label.color = "black",
  node.size = 30,
  node.color = "dodgerblue",
  ...
)
```

**Arguments**

- `x` an object of class `getEstimator` with `estimator="Gy"
- `PmaxResponse` cutpoint for thresholding the learning structure matrix of multiple response variables. Default is 0.5
- `PtrueResponse` true adjacency matrix for the structure of multiple response variables
- `name.responses` A vector for the node names. The default is "NA" only to show the locations. Value "auto" show the response names from the orginal data.
- `edge.weight` draw weighted edges after thresholding at 0.5. The defaul value "FALSE" is not to draw weighted edges
- `label.color` label color. Default is "black"
- `node.size` node size. Default is 30
- `node.color` node color. Default is "dodgerblue"
- `...` other arguments

**Examples**

```r
data("example_eQTL", package = "BayesSUR")
hyperpar <- list( a_w = 2 , b_w = 5 )
set.seed(9173)
fit <- BayesSUR(Y = example_eQTL["blockList" ][[1]],
X = example_eQTL["blockList" ][[2]],
```
predict.BayesSUR

```r
predict.BayesSUR <- function(object, newx, type = c("response", "coefficients", "nonzero"), Pmax = 0, 
                           ...)
```

### Description

Predict responses corresponding to the posterior mean of the coefficients, return posterior mean of coefficients or indices of nonzero coefficients of a "BayesSUR" class object.

### Usage

```r
## S3 method for class 'BayesSUR'
predict(
  object,
  newx,
  type = c("response", "coefficients", "nonzero"),
  Pmax = 0,
  ...
)
```

### Arguments

- `object` an object of class "BayesSUR"
- `newx` Matrix of new values for x at which predictions are to be made. Must be a matrix
- `type` Type of prediction required. Type "response" gives the fitted responses. Type "coefficients" computes the coefficients truncated the estimated coefficients based on thresholding the estimated latent indicator variable at Pmax. Type "nonzero" returns a list of the indices of the nonzero coefficients corresponding to the estimated latent indicator variable thresholding at Pmax
- `Pmax` threshold that truncates the estimated coefficients based on thresholding the estimated latent indicator variable. Default is 0.
- `...` other arguments

### Value

Predicted values extracted from an object of class "BayesSUR". If the BayesSUR specified data standardization, the fitted values are base based on standardized data.
Examples

data("example_eQTL", package = "BayesSUR")
hyperpar <- list(a_w = 2, b_w = 5)

set.seed(9173)
fit <- BayesSUR(Y = example_eQTL["blockList"][1],
X = example_eQTL["blockList"][2],
data = example_eQTL["data"], outFilePath = tempdir(),
nIter = 100, burnin = 50, nChains = 2, gammaPrior = "hotspot",
hyperpar = hyperpar, tmpFolder = "tmp/"
)

## check prediction
predict.val <- predict(fit, newx=example_eQTL["blockList"][2])

print.BayesSUR

print a short summary of the Bayesian Seemingly Unrelated Regressions Fits

Description

Print a short summary of a "BayesSUR" class object. It includes the argument matching information, number of selected predictors based on thresholding the posterior mean of the latent indicator variable at 0.5 by default.

Usage

## S3 method for class 'BayesSUR'
print(x, Pmax = 0.5, ...)

Arguments

x an object of class "BayesSUR"

Pmax threshold that truncates the estimated coefficients based on thresholding the estimated latent indicator variable. Default is 0.5

... other arguments

Value

Return a short summary from an object of class "BayesSUR", including the number of selected predictors with mPIP>Pmax and the expected log pointwise predictive density estimates (i.e., elpd.LOO and elpd.WAIC).
Examples

data("example_eQTL", package = "BayesSUR")
hyperpar = list( a_w = 2 , b_w = 5 )

set.seed(9173)
fit <- BayesSUR(Y = example_eQTL["blockList"][1],
X = example_eQTL["blockList"][2],
data = example_eQTL["data"], outFilePath = tempdir(),
nIter = 100, burnin = 50, nChains = 2, gammaPrior = "hotspot",
hyperpar = hyperpar, tmpFolder = "tmp/", output_CPO=TRUE)

## check output
# show the print information
print(fit)

summary.BayesSUR

summarizing Bayesian Seemingly Unrelated Regressions Fits

Description

Summary method for class "BayesSUR". It includes the argument matching information, Top predictors/responses on average mPIP across all responses/predictors, elpd estimates, MCMC specification, model specification and hyper-parameters. The summarized number of the selected variable corresponds to the posterior mean of the latent indicator variable thresholding at 0.5 by default.

Usage

## S3 method for class 'BayesSUR'
summary(object, Pmax = 0.5, ...)

Arguments

object an object of class "BayesSUR"
Pmax threshold that truncates the estimated coefficients based on thresholding the estimated latent indicator variable. Default is 0.5
...
other arguments

Value

Return a result summary from an object of class "BayesSUR", including the CPOs, number of selected predictors with mPIP>Pmax, top 10 predictors on average mPIP across all responses, top 10 responses on average mPIP across all predictors, Expected log pointwise predictive density (elpd) estimates, MCMC specification, model specification (i.e., covariance prior and gamma prior) and hyper-parameters.
Examples

data(example_eQTL, package = "BayesSUR")
hyperpar = list(a_w = 2, b_w = 5)

set.seed(9173)
fit <- BayesSUR(Y = example_eQTL["blockList"][[1]],
                X = example_eQTL["blockList"][[2]],
data = example_eQTL["data"], outFilePath = tempdir(),
nIter = 100, burnin = 50, nChains = 2, gammaPrior = "hotspot",
hyperpar = hyperpar, tmpFolder = "tmp/", output_CPO=TRUE)

## check output
# show the summary information
summary(fit)

Description

Indexes list of target genes corresponding the example_GDSC data set. It has two components representing the gene indexes of the MAPK/ERK pathway and BCR-ABL gene fusion in the example_GDSC data set.

Usage

targetGene

Format

An object of class list of length 2.

Examples

# Load the indexes of gene targets from the GDSC sample dataset
data("targetGene", package = "BayesSUR")
str(targetGene)

## Not run:
#===============
# This code below is to do preprocessing of GDSC data and obtain the complete dataset
# "example_GDSC.rda" above. The user needs load the datasets from
# ftp://ftp.sanger.ac.uk/pub4/cancerrxgene/releases/release-5.0/.
# But downloading and transforming the three used datasets below to *.csv files first.
#===============

requireNamespace("plyr", quietly = TRUE)
requireNamespace("data.table", quietly = TRUE)

features <- data.frame(read.csv("gdsc_en_input_w5.csv", head=T))
names.fea <- strsplit(rownames(features), "")
features <- t(features)
p <- c(13321, 13747-13321, 13818-13747)
Cell.Line <- rownames(features)
features <- data.frame(Cell.Line, features)

ic50_00 <- data.frame(read.csv("gdsc_drug_sensitivity_fitted_data_w5.csv", head=T))
ic50_0 <- ic50_00[,c(1,4,7)]
drug.id <- data.frame(read.csv("gdsc_tissue_output_w5.csv", head=T)][,c(1,3)]
drug.id2 <- drug.id[!duplicated(drug.id$drug.id),]
# delete drug.id=1066 since ID1066 and ID156 both correspond drug AZD6482,
# and no ID1066 in the "suppl.Data1" by Garnett et al. (2012)
drug.id2 <- drug.id2[drug.id2$drug.id!=1066,]
drug.id2$drug.name <- as.character(drug.id2$drug.name)
drug.id2$drug.name <- substr(drug.id2$drug.name, 1, nchar(drug.id2$drug.name)-6)
drug.id2$drug.name <- gsub(" ", ",", drug.id2$drug.name)

ic50 <- ic50_0
# mapping the drug_id to drug names in drug sensitivity data set
ic50$drug_id <- plyr::mapvalues(ic50$drug_id, from = drug.id2[,2], to = drug.id2[,1])

colnames(ic50) <- c("Cell.Line", "compound", "IC50")

# transform drug sensitivity overall cell lines to a data matrix
y0 <- reshape(ic50, v.names="IC50", timevar="compound", idvar="Cell.Line", direction="wide")
y0$Cell.Line <- gsub(" ", ",", y0$Cell.Line)

# select nonmissing pharmacological data

y00 <- y0
m0 <- dim(y0)[2]-1
eps <- 0.05
# r1.na is better to be not smaller than r2.na
r1.na <- 0.3
r2.na <- 0.2
k <- 1
while(sum(is.na(y0[,2:(1+m0)]))>0){
  r1.na <- r1.na - eps/k
  r2.na <- r1.na - eps/k
  k <- k + 1
  ## select drugs with <30% (decreasing with k) missing data overall cell lines
  na.y <- apply(y0[,2:(1+m0)], 2, function(xx) sum(is.na(xx))/length(xx))
  while(sum(na.y<r1.na)<m0){
    y0 <- y0[,-c(1+which(na.y>r1.na))]
    m0 <- sum(na.y<r1.na)
    na.y <- apply(y0[,2:(1+m0)], 2, function(xx) sum(is.na(xx))/length(xx))
  }
  ## select cell lines with treatment of at least 80% (increasing with k) drugs
  na.y0 <- apply(y0[,2:(1+m0)], 1, function(xx) sum(is.na(xx))/length(xx))
}
while(sum(na.y0<r2.na)<(dim(y0)[1])){
  y0 <- y0[na.y0<r2.na,]
  na.y0 <- apply(y0[,2:(1+m0)], 1, function(xx) sum(is.na(xx))/length(xx))
  num.na <- sum(is.na(y0[,2:(1+m0)]))
  message("#(NA)=", num.na, "\n", "r1.na =", r1.na, ", r2.na =", r2.na, "\n")
}

### combine drug sensitivity, tissues and molecular features

yx <- merge(y0, features, by="Cell.Line")
names.cell.line <- yx$Cell.Line
names.drug <- colnames(yx)[2:(dim(y0)[2])]
names.drug <- substr(names.drug, 6, nchar(names.drug))
# numbers of gene expression features, copy number feastures and mutation features
p <- c(13321, 13747-13321, 13818-13747)
num.nonpen <- 13
yx <- data.matrix(yx[, -1])
y <- yx[,1:(dim(y0)[2]-1)]
x <- cbind(yx[,dim(y0)[2]-1+sum(p)+1:num.nonpen], yx[,dim(y0)[2]-1+1:sum(p)])
# delete genes with only one mutated cell line
GDSC <- list(y=y, x=x, p=p, num.nonpen=num.nonpen, names.cell.line=names.cell.line,
names.drug=names.drug)

# select a small set of drugs

name_drugs <- c("Methotrexate","RDEA119","PD-0325901","CI-1040","AZD6244","Nilotinib",
"Axitinib")
# extract the drugs' pharmacological profiling and tissue dummy
# delete the cell line with extreme log(IC50)=-36.49 for drug "AP-24534"
col_filter <- colnames(GDSC$y) %in% paste("IC50.\"", name_drugs, sep="")
YX0 <- cbind(GDSC$y[, col_filter], c(1, 3, 6, 4, 7, 2, 5)),
GDSC$x[, 1:GDSC$num.nonpen]
)
colnames(YX0) <- c(name_drugs, colnames(GDSC$x)[1:GDSC$num.nonpen])
# extract the genetic information of CNV & MUT
colnames(X23)[1:p[2]] <- paste(substr(colnames(X23)[1:p[2]], 1,
  nchar(colnames(X23)[1:p[2]])-3), ",.CNV", sep="")
# locate all genes with CNV or MUT information
name_genes_duplicate <- c( substr(colnames(X23)[1:p[2]], 1, nchar(colnames(X23)[1:p[2]])-4),
name_genes <- name_genes_duplicate[!duplicated(name_genes_duplicate)]

# select the GEX which have the common genes with CNV or MUT
col_filter <- GDSC$num.nonpen + which(
  colnames(GDSC$x)[GDSC$num.nonpen+1:p[1]] %in% name_genes
)
X1 <- GDSC$x[, col_filter]
p[1] <- ncol(X1)
X1 <- log2(X1)

# summary the data information
eexample_GDSC <- list( data=cbind( YX0, X1, X23 ) )
eexample_GDSC$blockList <- list(1:length(name_drugs), length(name_drugs)+1:GDSC$num.nonpen,
  ncol(YX0)+1:sum(p))

# construct the G matrix: edge potentials in the MRF prior

# edges between drugs: Group1 ("RDEA119", "17-AAG", "PD-0325901", "CI-1040" and "AZD6244")
# indexed as (2:5)
# The MAPK_pathway.txt file is originally from KEGG or GSEA database at
# http://software.broadinstitute.org/gsea/msigdb/cards/KEGG_MAPK_SIGNALING_PATHWAY
pathway_genes <- read.table("MAPK_pathway.txt")[[1]]
X1_names_dup <- c(colnames(X1), name_genes_duplicate)
Idx_Pathway1 <- which(X1_names_dup %in% pathway_genes)
rep1 <- rep(Idx_Pathway1, each=length(2:5))
rep2 <- rep((2:5-1) * sum(p), times=length(Idx_Pathway1))
rep3 <- rep1 + rep2
Gmrf_Group1Pathway1 <- t(combn(rep3, 2))

# edges between drugs: Group2 ("Nilotinib", "Axitinib") indexed as (6:7)
# delete gene ABL2
Idx_Pathway2 <- which(X1_names_dup %like% "BCR" | X1_names_dup %like% "ABL")[-c(3,5)]
Gmrf_Group2Pathway2 <- t(combn(rep(Idx_Pathway2, each=length(6:7)) +
  rep((6:7-1)*sum(p), times=length(Idx_Pathway2)), 2))

# edges between the common gene in different data sources
Gmrf_CommonGene <- NULL
list_CommonGene <- list(0)
k <- 1
for(i in 1:length(name_genes)){
  Idx_CommonGene <- which(c(colnames(X1),name_genes_duplicate) == name_genes[i])
  if(length(Idx_CommonGene) > 1){
    Gmrf_CommonGene <- rbind(Gmrf_CommonGene,t(combn(Idx_CommonGene,each=length(name_drugs)) +
      rep((1:length(name_drugs)-1)*sum(p),times=length(Idx_CommonGene)), 2)))
    k <- k+1
  }
}
Gmrf_duplicate <- rbind( Gmrf_Group1Pathway1, Gmrf_Group2Pathway2, Gmrf_CommonGene )
Gmrf <- Gmrf_duplicate[!duplicated(Gmrf_duplicate),]
example_GDSC$mrfG <- Gmrf

# create the target gene names of the two groups of drugs
targetGenes1 <- matrix(Idx_Pathway1,nrow=1)
colnames(targetGenes1) <- colnames(example_GDSC$data)[seq_along(targetGene$group1)]
targetGenes2 <- matrix(Idx_Pathway2,nrow=1)
colnames(targetGenes2) <- colnames(example_GDSC$data)[seq_along(targetGene$group2)]

targetGene <- list(group1=targetGenes1, group2=targetGenes2)

## Write data file targetGene.rda to the user's directory by save()

## End(Not run)
Index

*Topic **datasets**
  
  example_eQTL, 9
  example_GDSC, 14
  targetGene, 34
  
BayesSUR, 2
BayesSUR_internal, 6
  
coef.BayesSUR, 7
  
elpd, 8
example_eQTL, 9
example_GDSC, 14
  
fitted.BayesSUR, 18
  
getEstimator, 19
  
plot.BayesSUR, 20
plot.CPO, 21
plot.Estimator, 23
plot.Manhattan, 25
plot.MCMCdiag, 26
plot.Network, 27
plot.ResponseGraph, 30
predict.BayesSUR, 31
print.BayesSUR, 32
  
summary.BayesSUR, 33
  
targetGene, 34

39