Package: bayesPO (via r-universe)

September 29, 2024

Type Package

Title Bayesian Inference for Presence-Only Data

Version 0.5.0

Date 2024-02-01

Contact Guido Alberti Moreira <guidoalber@gmail.com>

Maintainer Guido Alberti Moreira <guidoalber@gmail.com>

Description Presence-Only data is best modelled with a Point Process
Model. The work of Moreira and Gamerman (2022)
<doi:10.1214/21-AOAS1569> provides a way to use exact Bayesian
inference to model this type of data, which is implemented in
this package.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.1

Depends R (>= 3.5.0)

LinkingTo Rcpp, RcppEigen, RcppProgress

Imports Rcpp, coda, parallel, methods, RcppProgress, graphics, stats,
tools

Suggests bayesplot, knitr, rmarkdown, webshot, ggplot2, MASS

Collate 'RcppExports.R' 'bayesPO-package.R' 'prior-class.R'
'initial-class.R' 'model-class.R' 'fit-class.R' 'bayesPO.R'
'covariate_importance-class.R'

VignetteBuilder knitr

NeedsCompilation yes

Author Guido Alberti Moreira [cre, aut]
(<https://orcid.org/0000-0001-7557-0874>)

Repository CRAN

Date/Publication 2024-02-01 09:40:13 UTC

Contents

`bayesPO_fit-class` *Class for the result of the MCMC procedure.*

Description

Objects of this class are the main objects of this package. They contain much information about the fitted model.

Usage

```
## S4 method for signature 'bayesPO_fit'
show(object)

## S4 method for signature 'bayesPO_fit'
print(x, ...)

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

## S4 method for signature 'bayesPO_fit'
summary(object, ...)

## S3 method for class 'bayesPO_fit'
summary(object, ...)

## S4 method for signature 'bayesPO_fit'
names(x)
```

```
## S3 method for class 'bayesPO_fit'
names(x)

## S4 method for signature 'bayesPO_fit'
x[[i]]

## S4 method for signature 'bayesPO_fit'
x$name

## S4 method for signature 'bayesPO_fit'
as.array(x, ...)

## S3 method for class 'bayesPO_fit'
as.array(x, ...)

## S4 method for signature 'bayesPO_fit'
as.matrix(x, ...)

## S3 method for class 'bayesPO_fit'
as.matrix(x, ...)

## S4 method for signature 'bayesPO_fit'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'bayesPO_fit'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S4 method for signature 'bayesPO_fit,bayesPO_fit'
e1 + e2

## S4 method for signature 'bayesPO_fit'
c(x, ...)
```

Arguments

`object`	A bayesPO_fit object.
`x`	A bayesPO_fit object.
`...`	Ignored in this version.
`i`	The requested slot.
`name`	The requested slot.
`row.names`	NULL or a character vector giving the row names for the data frame. Missing values are not allowed.
`optional`	logical. If TRUE, setting row names and converting column names to syntactic names is optional. See help('as.data.frame') for more. Leaving as `FALSE` is recommended.
`e1`	A bayesPO_fit object.
`e2`	A bayesPO_fit object with the same slots (except for initial values) as `e1`.

Value

show and print: The invisible object.

summary: A matrix with the summary statistics of the fit. It is also printed in the print method. The summary can be treated as a matrix, such as retrieving rows/columns and creating tables with the xtable package.

names: A character vector with the available options for the `$` and `[[` methods.

'$' and '[[': The requested slot. Available options are not necessarily the class slots, and can be checked with the names method.

as.array: An array with dimensions I x C x P, where I stands for number of iterations, C for number of chains and P for total number of parameters. P is actually larger than the number of parameters in the model, as the the generated sizes of the latent processes and the log-posterior are also included. This is organized so that is ready for the bayesplot package functions.

as.matrix: The dimension of the output is I * C x (P + 2), where I stands for number of iterations, C for number of chains and P for total number of parameters. P is actually larger than the number of parameters in the model, as the generated sizes of the latent processes and the log-posterior are also included.

Two extra columns are included to indicate to which chain and to which iteration that draw belongs.

as.data.frame: The dimension of the output is I*C x P + 2, where I stands for number of iterations, C for number of chains and P for total number of parameters. P is actually larger than the number of parameters in the model, as the generated sizes of the latent processes and the log-posterior are also included.

Two extra columns are included to indicate to which chain and to which iteration that draw belongs. This is to facilitate the use of plotting results via the ggplot2 package if desired.

If row.names is left at NULL then row names are created as CcIi where c is the chain and i is the iteration of that row.

+: A new bayesPO_fit object where the chains are combined into a new multi-chain object. This can be used if chains are run in separate occasions or computers to combine them into a single object for analysis.

c: A new bayesPO_fit object where the chains are combined into a new multi-chain object. The + method is used for that, with the same arguments restrictions and results.

Fields

fit The actual fit from the model. It is an object of class mcmc.list, as generated from the coda package.

original The model used to generate the chains, an object with class bayesPO_model.

backgroundSummary A small summary of the original background covariates. This is to ensure that continuing the chains will use the identical background matrix. Only the summary is kept for storage efficiency.

area A positive number indicating the area measure of the region being studied.

parnames The names of the parameters. If the model used selects the covariates with column names, they are replicated here. If they are the column indexes, names are generated for identification.

mcmc_setup The original mcmc setup used.

See Also

fit_bayesPO

`bayesPO_initial-class` *Class for the initial values for the MCMC for the bayesPO package*

Description

Class for the initial values for the MCMC for the bayesPO package

Usage

```
## S4 method for signature 'bayesPO_initial'
names(x)

## S4 method for signature 'bayesPO_initial'
x$name

## S4 method for signature 'bayesPO_initial,ANY'
e1 + e2

## S4 method for signature 'list,bayesPO_initial'
e1 + e2

## S4 method for signature 'bayesPO_initial,list'
e1 + e2

## S4 method for signature 'bayesPO_initial,numeric'
e1 * e2

## S4 method for signature 'numeric,bayesPO_initial'
e1 * e2

## S4 method for signature 'bayesPO_initial'
show(object)

## S4 method for signature 'bayesPO_initial'
print(x, ...)

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

Arguments

`x`	The bayesPO_initial object.
`name`	The requested slot.

`e1`	A bayesPO_initial object.
`e2`	Another bayesPO_initial object or a list with bayesPO_initial objects for **+** and a positive integer for *****. e1 and e2 can be switched (+ and * are commutative).
`object`	A bayesPO_initial object.
`...`	Currently unused.

Value

`names`: A character vector with the initialized parameter names.

`‘$‘`: The requested initial value (in case of LambdaStar) or values (in case of Beta or Delta).

`+`: A list with the objects. Useful to start the `fit_bayesPO` function, as it requires a list of initial values.

`*`: A list with `e2` random initial values.

`show` and `print`: The invisible object.

Fields

`beta` Initial values for beta.

`delta` Initial values for delta.

`lambdaStar` Initial values for lambdaStar.

`tag` Indicates the source of the initial values.

`bayesPO_model` *Build a model to be used in the* `bayesPO` *fitting function*

Description

Constructor for `bayesPO_model-class` objects, built to facilitate the use of the fitting function. The output of this function has the necessary signature for the fit_bayesPO function to start the model fit.

Usage

```
bayesPO_model(
  po,
  intensitySelection,
  observabilitySelection,
  intensityLink = "logit",
  observabilityLink = "logit",
  initial_values = 1,
 joint_prior = prior(beta = NormalPrior(rep(0, length(intensitySelection) + 1), 10 *
    diag(length(intensitySelection) + 1)), delta = NormalPrior(rep(0,
   length(observabilitySelection) + 1), 10 * diag(length(observabilitySelection) + 1)),
    lambdaStar = GammaPrior(1e-10, 1e-10)),
  verbose = TRUE
)
```

Arguments

`po`	A matrix whose rows represent the presence-only data and the columns the covariates observed at each position.
`intensitySelection`	Either a numeric or character vector and represents the selection of covariates used for the intensity set. If numeric it is the positions of the columns and if character, the names of the columns.
`observabilitySelection`	Either a numeric or character vector and represents the selection of covariates used for the observability set. If numeric it is the positions of the columns and if character, the names of the columns.
`intensityLink`	A string to inform what link function the model has with respect to the intensity covariates. Current version accepts 'logit'.
`observabilityLink`	A string to inform what link function the model has with respect to the observabilitycovariates. Current version accepts 'logit'.
`initial_values`	Either a single integer, a single `bayesPO_initial-class` or a list containing `bayesPO_initial-class` objects. The length of the list will inform the model how many independent chains will be run. If an integer, that many initial values will be randomly generated.
`joint_prior`	A `bayesPO_prior` object.
`verbose`	Set to `FALSE` to suppress all messages to console.

Value

A `bayesPO_model` object with the requested slots. It is ready to be used in the `fit_bayesPO` function.

See Also

`initial`, `prior` and `fit_bayesPO`.

Examples

```
# Let us simulate some data to showcase the creation of the model.
beta <- c(-1, 2)
delta <- c(3, 4)
lambdaStar <- 1000

total_points <- rpois(1, lambdaStar)
random_points <- cbind(runif(total_points), runif(total_points))

# Find covariate values to explain the species occurrence.
# We give them a Gaussian spatial structure.
Z <- MASS::mvrnorm(1, rep(0, total_points), 3 * exp(-as.matrix(dist(random_points)) / 0.2))

# Thin the points by comparing the retaining probabilities with uniforms
# in the log scale to find the occurrences
```

```
occurrences <- log(runif(total_points)) <= -log1p(exp(-beta[1] - beta[2] * Z))
n_occurrences <- sum(occurrences)
occurrences_points <- random_points[occurrences,]
occurrences_Z <- Z[occurrences]

# Find covariate values to explain the observation bias.
# Additionally create a regular grid to plot the covariate later.
W <- MASS::mvrnorm(1, rep(0, n_occurrences), 2 * exp(-as.matrix(dist(occurrences_points)) / 0.3))

# Find the presence-only observations.
po_sightings <- log(runif(n_occurrences)) <= -log1p(exp(-delta[1] - delta[2] * W))
n_po <- sum(po_sightings)
po_points <- occurrences_points[po_sightings, ]
po_Z <- occurrences_Z[po_sightings]
po_W <- W[po_sightings]

# Now we create the model
model <- bayesPO_model(po = cbind(po_Z, po_W),
  intensitySelection = 1, observabilitySelection = 2,
  intensityLink = "logit", observabilityLink = "logit",
  initial_values = 2, joint_prior = prior(
    NormalPrior(rep(0, 2), 10 * diag(2)),
    NormalPrior(rep(0, 2), 10 * diag(2)),
    GammaPrior(1e-4, 1e-4)))
# Check how it is.
model
```

---

bayesPO_model-class *Class that defines a model for the bayesPO package.*

Description

The model includes the presence-only data, all selected variables, the link functions for q and p, the initial values and the prior distribution.

Usage

```
## S4 method for signature 'bayesPO_model'
names(x)

## S4 method for signature 'bayesPO_model'
x$name

## S4 replacement method for signature 'bayesPO_model'
x$name <- value

## S4 method for signature 'bayesPO_model'
show(object)
```

```
## S4 method for signature 'bayesPO_model'
print(x, ...)

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

Arguments

`x`	The bayesPO_model object.
`name`	The requested slot.
`value`	New value.
`object`	The bayesPO_model object.
`...`	Currently unused.

Value

`names`: A character vector with possible options for the `` `$` `` and `` `$<-` `` methods.

‘`$`‘: The requested slot’s value.

‘`$<-`‘: The new object with the updated slot.

`show` and `print`: The invisible object.

Fields

`po` The matrix containing the covariates values for the data.

`intensityLink` A string informing about the chosen link for the intensity covariates. Current acceptable choice is only `"logit"`.

`intensitySelection` A vector containing the indexes of the selected intensity columns in the `po` matrix.

`observabilityLink` A string informing about the chosen link for the observability covariates. Current acceptable choice is only `"logit"`.

`observabilitySelection` A vector containing the indexes of the selected observability columns in the `po` matrix.

`init` A list with objects of class `bayesPO_initial` indicating the initial values for each chain. The length of this list tells the program how many chains are requested to be run.

`prior` An object of class `bayesPO_prior` which indicates the joint prior distribution for the model parameters.

`iSelectedColumns` If the intensity covariates selection was made with the name of the columns, they are stored in this slot.

`oSelectedColumns` If the observability covariates selection was made with the name of the columns, they are stored in this slot.

See Also

`bayesPO_initial-class` and `bayesPO_prior-class` and `bayesPO_model`

`bayesPO_prior-class` *Joint prior class for the bayesPO package parameters*

Description

Objects of this class are the joining of independent priors for Beta, Delta and LambdaStar. They can be used in the `fit_bayesPO` function.

Usage

```
## S4 method for signature 'bayesPO_prior'
names(x)

## S4 method for signature 'bayesPO_prior'
x$name

## S4 method for signature 'bayesPO_prior'
show(object)

## S4 method for signature 'bayesPO_prior'
print(x, ...)

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

## S4 method for signature 'bayesPO_prior'
x$name

## S4 replacement method for signature 'bayesPO_prior'
x$name <- value
```

Arguments

`x`	The bayesPO_prior object.
`name`	The requested slot.
`object`	The bayesPO_prior object.
`...`	Ignored.
`value`	New value.

Value

`names`: A character vector with the model parameters names.

`` `$` ``: The requested slot's value.

`` `$<-` ``: The new object with the updated slot.

Fields

`beta` An object of a class which inherits the `BetaDeltaPrior` S4 class with the appropriate Beta prior.

`delta` An object of a class which inherits the `BetaDeltaPrior` S4 class with the appropriate Delta prior.

`lambdaStar` An object of a class which inherits the `LambdaStarPrior` S4 class with the appropriate LambdaStar prior.

`BetaDeltaPrior-class` *Generic class for the beta and delta parameters.*

Description

Generic class for the beta and delta parameters.

Usage

```
## S4 method for signature 'BetaDeltaPrior'
show(object)

## S4 method for signature 'BetaDeltaPrior'
print(x, ...)

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

Arguments

`object`	The BetaDeltaPrior object.
`x`	The BetaDeltaPrior object.
`...`	Ignored.

Value

`show` and `print`: The invisible object.

Fields

`family` The family of distributions of the prior.

covariates_importance-class
Class for covariates importance matrices

Description

Objects of this class is the output of the "covariates_importance" object from the `bayesPO_fit-class`. It can be plotted which uses the `graphics` package. The `print` method gives a point-wise estimation, the same seen in the `bacplot` method. Both `plot` and `boxplot` methods use the posterior distribution of the importance.

Usage

```
## S3 method for class 'covariates_importance'
print(x, component = "intensity", ...)

## S3 method for class 'covariates_importance'
plot(
  x,
  component = "intensity",
  y = "importance",
  quantiles = c(0.025, 0.5, 0.975),
  ...
)

## S3 method for class 'covariates_importance'
barplot(height, component = "intensity", y, ...)

## S3 method for class 'covariates_importance'
boxplot(x, component = "intensity", ...)
```

Arguments

`x`	The `covariates_importance` object.
`component`	Either `"intensity"`, `"observability"` or `"both"`.
`...`	Other parameters passed to `boxplot`.
`y`	Either `"interval"` or `"density"`. The formal gives vertical credible intervals, and the latter gives separate density plots with the specified quantiles as vertical lines.
`quantiles`	A 2- or 3-simensional vector with the desired quantiles specified. If 3-dimensiona, the middle point is drawn as a dot when the `y` parameter is set as `"interval"`.
`height`	The `covariates_importance` object.

Details

Objects of this class have two matrices where the Monte Carlo samples on the rows and parameters on the columns. One matrix is for the intensity importance and the other for the observability importance.

Value

The invisible object.

Nothing is returned. Plot is called and drawn on the configured device.

A barplot. See `barplot` for details. If component is selected as `"both"`, only the second barplot is returned.

A boxplot. See `boxplot` for details. If component is selected as `"both"`, only the second boxplot is returned.

See Also

`barplot`.

`boxplot`.

`fit_bayesPO` *Fit presence-only data using a Bayesian Poisson Process model*

Description

The model uses a data augmentation scheme to avoid performing approximations on the likelihood function.

Usage

```
fit_bayesPO(
  object,
  background,
  mcmc_setup = list(iter = 5000),
  verbose = TRUE,
  ...
)

## S4 method for signature 'bayesPO_model,matrix'
fit_bayesPO(
  object,
  background,
  mcmc_setup,
  verbose = TRUE,
  area = 1,
  cores = 1,
  ...
)

## S4 method for signature 'bayesPO_fit,matrix'
fit_bayesPO(
  object,
  background,
  mcmc_setup = list(iter = object$mcmc_setup$iter),
  verbose = TRUE,
  cores = 1,
  ...
)
```

Arguments

object	Either a bayesPO_model or bayesPO_fit object. If a model, then the model is fit according to specifications. If a fit, then the model used to fit the model is recovered and used to continue the MCMC calculations where the previous one left off.
background	A matrix where the rows are the grid cells for the studied region and the columns are the covariates. NAs must be removed. If the function is being used on a bayesPO_fit object, the background must be exactly the same as the one used in the original fit.
mcmc_setup	A list containing iter to inform the model how many iterations are to be run. The list may optionally contain the objects.
verbose	Set to FALSE to suppress all messages to console.
...	Parameters passed on to specific methods. burnin and thin to inform these instructions as well.
area	A positive number with the studied region's area.
cores	Currently unused.

Details

The background is kept outside of the

Value

An object of class "bayesPO_fit".

See Also

bayesPO_model and bayesPO_fit-class.

Examples

```
# This code is replicated from the vignette.
## Not run:
beta <- c(-1, 2) # Intercept = -1. Only one covariate
```

```
delta <- c(3, 4) # Intercept = 3. Only one covariate
lambdaStar <- 1000

total_points <- rpois(1, lambdaStar)
random_points <- cbind(runif(total_points), runif(total_points))
grid_size <- 50

# Find covariate values to explain the species occurrence.
# We give them a Gaussian spatial structure.
reg_grid <- as.matrix(expand.grid(seq(0, 1, len = grid_size), seq(0, 1, len = grid_size)))
Z <- MASS::mvrnorm(1, rep(0, total_points + grid_size * grid_size),
  3 * exp(-as.matrix(dist(rbind(random_points, reg_grid))) / 0.2))
Z1 <- Z[1:total_points]; Z2 <- Z[-(1:total_points)]

# Thin the points by comparing the retaining probabilities with uniforms
# in the log scale to find the occurrences
occurrences <- log(runif(total_points)) <= -log1p(exp(-beta[1] - beta[2] * Z1))
n_occurrences <- sum(occurrences)
occurrences_points <- random_points[occurrences,]
occurrences_Z <- Z1[occurrences]

# Find covariate values to explain the observation bias.
# Additionally create a regular grid to plot the covariate later.
W <- MASS::mvrnorm(1, rep(0, n_occurrences + grid_size * grid_size),
  2 * exp(-as.matrix(dist(rbind(occurrences_points, reg_grid))) / 0.3))
W1 <- W[1:n_occurrences]; W2 <- W[-(1:n_occurrences)]

# Find the presence-only observations.
po_sightings <- log(runif(n_occurrences)) <= -log1p(exp(-delta[1] - delta[2] * W1))
n_po <- sum(po_sightings)
po_points <- occurrences_points[po_sightings, ]
po_Z <- occurrences_Z[po_sightings]
po_W <- W1[po_sightings]

jointPrior <- prior(
  NormalPrior(rep(0, 2), 10 * diag(2)), # Beta
NormalPrior(rep(0, 2), 10 * diag(2)), # Delta
GammaPrior(0.00001, 0.00001) # LambdaStar
)

model <- bayesPO_model(po = cbind(po_Z, po_W),
intensitySelection = 1, observabilitySelection = 2,
                    intensityLink = "logit", observabilityLink = "logit",
                    initial_values = 2, joint_prior = jointPrior)

bkg <- cbind(Z2, W2) # Create background

fit <- fit_bayesPO(model, bkg, area = 1, mcmc_setup = list(burnin = 1000, iter = 2000))

summary(fit)

# Rhat upper CI values are above 1.1. More iterations are needed, so...
```

```
fit2 <- fit_bayesPO(fit, bkg, mcmc_setup = list(iter = 10000))

summary(fit2)
mcmc_trace(fit2)
mcmc_dens(fit2)

## End(Not run)
```

GammaPrior — *Create a Gamma prior object for model specification.*

Description

Constructor for `GammaPrior-class` objects

Usage

```
GammaPrior(shape, rate)
```

Arguments

`shape`	A positive number.
`rate`	A positive number.

Value

A `GammaPrior` object with adequate slots.

GammaPrior-class — *Gamma prior class for the LambdaStar parameter.*

Description

This is used to represent the prior for lambdaStar individually. It still needs to be joined with the prior for Beta and Delta to be used in a model.

Usage

```
## S4 method for signature 'GammaPrior'
names(x)

## S4 method for signature 'GammaPrior'
x$name

## S4 replacement method for signature 'GammaPrior'
x$name <- value
```

```
## S4 method for signature 'GammaPrior'
show(object)

## S4 method for signature 'GammaPrior'
print(x, ...)

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

Arguments

`x`	The GammaPrior object.
`name`	The requested slot.
`value`	New value.
`object`	The GammaPrior object.
`...`	Ignored.

Value

`names`: A character vector with the prior parameters.

`‘$‘` The requested slot’s value.

`‘$<-‘`: The new object with the updated slot.

`show` and `print`: The invisible object.

Fields

`shape` The shape parameter of the Gamma distribution.

`rate` The rate parameter of the Gamma distribution.

See Also

`prior`

Examples

```
GammaPrior(0.0001, 0.0001)
```

initial *Initial values constructor for bayesPO modeling*

Description

Helper function to create a valid set of initial values to be used with the fit_bayesPO function.

Usage

```
initial(
  beta = numeric(),
  delta = numeric(),
  lambdaStar = numeric(),
  random = FALSE
)
```

Arguments

`beta`	Either a vector or a single integer. The vector is used if the initial values are provided and the integer is used as the vector size to be randomly generated.
`delta`	Either a vector or a single integer. The vector is used if the initial values are provided and the integer is used as the vector size to be randomly generated.
`lambdaStar`	A positive number.
`random`	A logical value. If `TRUE`, then the initial values are generated from standard normal distribution for `beta` and `delta` and from a `Beta(lambdaStar, 1)` for `lambdaStar`. The latter is generated as a low value due to potential explosive values resulting from background area scaling.

Value

A `bayesPO_initial` object. It can be used in the `fit_bayesPO` function by itself, but must be in a list if multiple initial values are supplied. Initial values can be combined by adding them (with the use of '+').

See Also

`bayesPO_initial-class`.

Examples

```
# Let us create initial values for a model with, say, 3 intensity covariates
# and 4 observability covariates. We add an initial values for both these
# cases due to the intercepts.

# This first one is
in1 <- initial(rep(0, 4), c(0, 2, -1, -2, 3), 100)
```

```
# Then we initalize some randomly.
in2 <- initial(4, 5, 100, random = TRUE)

# We can even multiply the random one to generate more. Let us join them all
# to include in a model.
initial_values <- in1 + in2 * 3
# 4 chains are initialized.
```

`LambdaStarPrior-class` *Generic class for the LambdaStar parameters.*

Description

Generic class for the LambdaStar parameters.

Usage

```
## S4 method for signature 'LambdaStarPrior'
show(object)
```

Arguments

`object`	The LambdaStarPrior object.

Value

`show` and `print`: The invisible object.

Fields

`family` The family of distributions of the prior.

`NormalPrior` *Create a Normal prior object for model specification.*

Description

Constructor for `NormalPrior-class` objects

Usage

```
NormalPrior(mu, Sigma)
```

Arguments

`mu`	The mean vector for the Normal distribution.
`Sigma`	The covariance matrix for the Normal distribution.

Details

Matrix Sigma must be square and positive definite. Its dimensions must match mu's length.

Value

A `NormalPrior` object with adequate slots.

See Also

`prior`

Examples

```
NormalPrior(rep(0, 10), diag(10) * 10)
```

`NormalPrior-class` *Normal prior class for Beta and Delta parameters.*

Description

This is used to represent the prior for Beta and Delta individually. They still need to be joined to be used in a model.

Usage

```
## S4 method for signature 'NormalPrior'
names(x)

## S4 method for signature 'NormalPrior'
x$name

## S4 replacement method for signature 'NormalPrior'
x$name <- value

## S4 method for signature 'NormalPrior'
show(object)

## S4 method for signature 'NormalPrior'
print(x, ...)

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

Arguments

`x`	The NormalPrior object.
`name`	The requested slot.
`value`	New value.
`object`	The NormalPrior object.
`...`	Ignored.

Value

`names`: A character vector with the prior parameters.

`‘$‘`: The requested slot’s value.

`‘$<-‘`: The new object with the updated slot.

`show` and `print`: The invisible object.

Fields

`mu` The mean vector for the prior.

`Sigma` The covariance matrix for the prior.

`prior` *Build a joint prior for bayesPO model parameters*

Description

Constructor for `bayesPO_prior` objects, which is used in the `bayesPO_fit` function. The generated prior is so that Beta, Delta and LambdaStar are indepdendent a priori.

Usage

```
prior(beta, delta, lambdaStar)
```

Arguments

`beta`	An S4 object whose class inherits from `BetaDeltaPrior`.
`delta`	An S4 object whose class inherits from `BetaDeltaPrior`.
`lambdaStar`	An S4 object whose class inherits from `LambdaStarPrior`.

Value

A `bayesPO_prior` object with the adequate slots. It is ready to be included in a model via the `bayesPO_model` function.

See Also

`fit_bayesPO`, `NormalPrior`, `GammaPrior` and `bayesPO_model`.

Examples

```
# Let us say there are 3 intensity covariates and 4 observability covariates.
# One more element is included in both sets due to the intercepts.
new_prior <- prior(
  NormalPrior(rep(0, 4), 10 * diag(4)),
  NormalPrior(rep(0, 5), 10 * diag(5)),
  GammaPrior(0.0001, 0.0001)
)
```

Index