Package: AntMAN (via r-universe)

October 1, 2024

Version 1.1.0

Date 2021-07-01

Type Package

Title Anthology of Mixture Analysis Tools

Author Priscilla Ong [aut, edt], Raffaele Argiento [aut], Bruno Bodin [aut, cre], Maria De Iorio [aut]

Maintainer Bruno Bodin <bruno.bodin@yale-nus.edu.sg>

Description Fits finite Bayesian mixture models with a random number of components. The MCMC algorithm implemented is based on point processes as proposed by Argiento and De Iorio (2019) <arXiv:1904.09733> and offers a more computationally efficient alternative to reversible jump. Different mixture kernels can be specified: univariate Gaussian, multivariate Gaussian, univariate Poisson, and multivariate Bernoulli (latent class analysis). For the parameters characterising the mixture kernel, we specify conjugate priors, with possibly user specified hyper-parameters. We allow for different choices for the prior on the number of components: shifted Poisson, negative binomial, and point masses (i.e. mixtures with fixed number of components).

License MIT + file LICENSE

LazyData true

URL https://github.com/bbodin/AntMAN

Imports stats, graphics, grDevices, Rcpp (>= 0.12.3), salso, mvtnorm, mcclust, GGally, bayesplot, Rdpack

RdMacros Rdpack

Suggests dendextend, ggdendro, ggplot2, jpeg

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.1.1

Encoding UTF-8

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-07-23 10:00:02 UTC

Contents

`AM_clustering` *Return the clustering matrix*

Description

Given an `AM_mcmc_output` object, this function returns the clustering matrix.

Usage

```
AM_clustering(fit)
```

Arguments

`fit`	an `AM_mcmc_output` object.

Details

The clustering matrix is an M by n matrix. Each of the M rows represents a clustering of n items using cluster labels. Items i and j are in the same cluster if fit[m,i] == fit[m,j] for the mth clustering.

Value

A numeric clustering matrix

See Also

`AM_coclustering`

Examples

```
fit = AM_demo_uvp_poi()$fit
ccm <- AM_clustering(fit)
```

AM_coclustering *Return the co-clustering matrix*

Description

Given an AM_mcmc_output object, this function returns the co-clustering matrix.

Usage

```
AM_coclustering(fit)
```

Arguments

fit	an AM_mcmc_output object.

Details

The co-clustering matrix is produced by the simultaneous clustering of the rows and columns. Each entry denotes the (posterior) probability that items i and j are together. This technique is also known as bi-clustering and block clustering (Govaert and Nadif 2013), and is useful for understanding the number of clusters in the dataset.

Value

A numeric co-clustering matrix

See Also

AM_clustering

Examples

```
fit = AM_demo_uvp_poi()$fit
ccm <- AM_coclustering(fit)
```

AM_demo_mvb_poi *Returns an example of AM_mcmc_fit output produced by the multivariate bernoulli model*

Description

This function allows us to generate a sample output of fitting the multivariate Bernoulli model. No arguments are needed to be passed. The purpose of this function is to serve as a demo for users to understand the model's output, without diving too deep into details. By default, this demo generates a sample dataset of dimension 500x4, where the MCMC sampler is specified to run for 2000 iterations, with a burn-in of 1000, and a thinning interval of 10. All possible outputs that can be produced by AM_mcmc_fit are returned (see return value below).

Usage

```
AM_demo_mvb_poi()
```

Value

A list containing the following items:

- the vector (or matrix) containing the synthetic data used to fit the model.
- the vector containing the final cluster assignment of each observation.
- an AM_mcmc_output object, which is the typical output of AM_mcmc_fit.

Examples

```
mvb_output <- AM_demo_mvb_poi()
```

AM_demo_mvn_poi *Returns an example of AM_mcmc_fit output produced by the multivariate gaussian model*

Description

This function allows us to generate a sample output of fitting the multivariate Gaussian model. No arguments are needed to be passed. The purpose of this function is to serve as a demo for users to understand the model's output, without diving too deep into details. By default, this demo generates a sample dataset of dimension 500x2, where the MCMC sampler is specified to run for 2000 iterations, with a burn-in of 1000, and a thinning interval of 10. All possible outputs that can be produced by AM_mcmc_fit are returned (see return value below).

Usage

```
AM_demo_mvn_poi()
```

Value

A list containing the following items:

- the vector (or matrix) containing the synthetic data used to fit the model.
- the vector containing the final cluster assignment of each observation.
- an AM_mcmc_output object, which is the typical output of AM_mcmc_fit.

Examples

```
mvn_output <- AM_demo_mvn_poi()
```

AM_demo_uvn_poi *Returns an example of* AM_mcmc_fit *output produced by the univariate Gaussian model*

Description

This function allows us to generate a sample output of fitting the univariate gaussian model. No arguments are needed to be passed. The purpose of this function is to serve as a demo for users to understand the model's output, without diving too deep into details. By default, this demo generates a sample dataset of dimension 500x1, where the MCMC sampler is specified to run for 2000 iterations, with a burn-in of 1000, and a thinning interval of 10. All possible outputs that can be produced by AM_mcmc_fit are returned (see return value below).

Usage

```
AM_demo_uvn_poi()
```

Value

A list containing the following items:

- the vector (or matrix) containing the synthetic data used to fit the model.
- the vector containing the final cluster assignment of each observation.
- an AM_mcmc_output object, which is the typical output of AM_mcmc_fit.

Examples

```
mvn_output <- AM_demo_uvn_poi()
```

AM_demo_uvp_poi *Returns an example of* AM_mcmc_fit *output produced by the univariate Poisson model*

Description

This function allows us to generate a sample output of fitting the univariate poisson model. No arguments are needed to be passed. The purpose of this function is to serve as a demo for users to understand the model's output, without diving too deep into details. By default, this demo generates a sample dataset of dimension 500x1, where the MCMC sampler is specified to run for 2000 iterations, with a burn-in of 1000, and a thinning interval of 10. All possible outputs that can be produced by AM_mcmc_fit are returned (see return value below).

Usage

```
AM_demo_uvp_poi()
```

Value

A list containing the following items:

- the vector (or matrix) containing the synthetic data used to fit the model.
- the vector containing the final cluster assignment of each observation.
- an AM_mcmc_output object, which is the typical output of AM_mcmc_fit.

Examples

```
mvn_output <- AM_demo_uvn_poi()
```

AM_emp_bayes_uninorm *compute the hyperparameters of an Normal-Inverse-Gamma distribution using an empirical Bayes approach*

Description

This function computes the hyperparameters of a Normal Inverse-Gamma distribution using an empirical Bayes approach. More information about how these hyperparameters are determined can be found here: *Bayes and empirical Bayes: do they merge?* (Petrone et al. 2012).

Usage

```
AM_emp_bayes_uninorm(y, scEmu = 1, scEsig2 = 3, CVsig2 = 3)
```

Arguments

y	The data y. If y is univariate, a vector is expected. Otherwise, y should be a matrix.
scEmu	a positive value (default=1) such that marginally E(μ) = s^2*scEmu, where s^2 is the sample variance.
scEsig2	a positive value (default=3) such that marginally E(σ^2) = s^2*scEsig2, where s^2 is the sample variance.
CVsig2	The coefficient of variation of σ^2 (default=3).

Value

an object of class AM_mix_hyperparams, in which hyperparameters m0, k0, nu0 and sig02 are specified. To understand the usage of these hyperparameters, please refer to AM_mix_hyperparams_uninorm.

AM_extract *Extract values within a* AM_mcmc_output *object*

Description

Given an AM_mcmc_output object, as well as the target variable names, AM_extract will return a list of the variables of interest.

Usage

```
AM_extract(object, targets, iterations = NULL, debug = FALSE)
```

Arguments

object	an AM_mcmc_output object.
targets	List of variables to extract (ie. K, M, mu).
iterations	Can specify particular iterations to extracts, NULL for all.
debug	Activate log to.

Details

Due to the complexity of AntMAN outputs, AM_mcmc_output object can be difficult to handle. The AM_extract function eases access of particular variables within the AM_mcmc_output object. Variables of varying dimension are expected to result from the transdimensional moves. When considering such variables, the extracted list would correspond to an nx1 list, where n refers to the number of extracted iterations. Each of these nx1 entries consists of another list of dimension mx1, where m specifies the number of components inferred for that iteration.

Value

a list of variables specified in targets.

AM_find_gamma_Delta *Given that the prior on M is a dirac delta, find the γ hyperparameter of the weights prior to match $E(K) = K*$, where $K*$ is user-specified*

Description

Once a fixed value of the number of components M^* is specified, this function adopts a *bisection method* to find the value of γ such that the induced distribution on the number of clusters is centered around a user specifed value K^*, i.e. the function uses a bisection method to solve for γ (Argiento and Iorio 2019). The user can provide a lower γ_l and an upper γ_u bound for the possible values of γ. The default values are $\gamma_l = 10^{-3}$ and $\gamma_u = 10$. A default value for the tolerance is $\epsilon = 0.1$. Moreover, after a maximum number of iteration (default is 31), the function stops warning that convergence has not been reached.

Usage

```
AM_find_gamma_Delta(
  n,
  Mstar,
  Kstar = 6,
  gam_min = 1e-04,
  gam_max = 10,
  tolerance = 0.1
)
```

Arguments

n	sample size.
Mstar	number of components of the mixture.
Kstar	mean number of clusters the user wants to specify.
gam_min	lower bound of the interval in which gamma should lie (default 1e-4).
gam_max	upper bound of the interval in which gamma should lie (default 10).
tolerance	Level of tolerance for the method.

Value

A value of gamma such that $E(K) = K^*$

Examples

```
n <- 82
Mstar <- 12
gam_de <- AM_find_gamma_Delta(n,Mstar,Kstar=6, gam_min=1e-4,gam_max=10, tolerance=0.1)
prior_K_de <-  AM_prior_K_Delta(n,gam_de,Mstar)
prior_K_de%*%1:n
```

AM_find_gamma_NegBin *Given that the prior on M is a Negative Binomial, find the γ hyper-parameter of the weights prior to match $E(K) = K*$, where $K*$ is user-specified*

Description

Once the prior on the number of mixture components M is assumed to be a Negative Binomial with parameter r>0 and 0<p<1, with mean is 1+ r*p/(1-p), this function adopts a *bisection method* to find the value of gamma such that the induced distribution on the number of clusters is centered around a user specifed value K^*, i.e. the function uses a bisection method to solve for γ (Argiento and Iorio 2019). The user can provide a lower γ_l and an upper γ_u bound for the possible values of γ. The default values are $\gamma_l = 10^{-3}$ and $\gamma_u = 10$. A defaault value for the tolerance is $\epsilon = 0.1$. Moreover, after a maximum number of iteration (default is 31), the function stops warning that convergence has not bee reached.

Usage

```
AM_find_gamma_NegBin(
  n,
  r,
  p,
  Kstar = 6,
  gam_min = 0.001,
  gam_max = 10000,
  tolerance = 0.1
)
```

Arguments

n	The sample size.
r	The dispersion parameter r of the Negative Binomial.
p	The probability of failure parameter p of the Negative Binomial.
Kstar	The mean number of clusters the user wants to specify.
gam_min	The lower bound of the interval in which gamma should lie.
gam_max	The upper bound of the interval in which gamma should lie.
tolerance	Level of tolerance of the method.

Value

A value of gamma such that $E(K) = K^*$

Examples

```
n <- 82
r <- 1
p <- 0.8571
gam_nb= AM_find_gamma_NegBin(n,r,p,Kstar=6, gam_min=0.001,gam_max=10000, tolerance=0.1)
prior_K_nb= AM_prior_K_NegBin(n,gam_nb, r, p)
prior_K_nb%*%1:n
```

AM_find_gamma_Pois *Given that the prior on M is a shifted Poisson, find the γ hyperparameter of the weights prior to match $E(K) = K\hat{}*$, where $K\hat{}*$ is user-specified*

Description

Once the prior on the number of mixture components M is assumed to be a Shifted Poisson of parameter Lambda, this function adopts a *bisection method* to find the value of γ such that the induced distribution on the number of clusters is centered around a user specifed value K^*, i.e. the function uses a bisection method to solve for γ (Argiento and Iorio 2019). The user can provide a lower γ_l and an upper γ_u bound for the possible values of γ. The default values are $\gamma_l = 10^{-3}$ and $\gamma_u = 10$. A defaault value for the tolerance is $\epsilon = 0.1$. Moreover, after a maximum number of iteration (default is 31), the function stops warning that convergence has not bee reached.

Usage

```
AM_find_gamma_Pois(
  n,
  Lambda,
  Kstar = 6,
  gam_min = 1e-04,
  gam_max = 10,
  tolerance = 0.1
)
```

Arguments

n	The sample size.
Lambda	The parameter of the Shifted Poisson for the number of components of the mixture.
Kstar	The mean number of clusters the user wants to specify.
gam_min	The lower bound of the interval in which gamma should lie.
gam_max	The upper bound of the interval in which gamma should lie.
tolerance	Level of tolerance of the method.

Value

A value of gamma such that $E(K) = K^*$

Examples

```
n <- 82
Lam  <- 11
gam_po <-  AM_find_gamma_Pois(n,Lam,Kstar=6, gam_min=0.0001,gam_max=10, tolerance=0.1)
prior_K_po <-  AM_prior_K_Pois(n,gam_po,Lam)
prior_K_po%*%1:n
```

AM_mcmc_configuration *S3 class AM_mcmc_configuration*

Description

Output type of return values from AM_mcmc_parameters.

Value

AM_mcmc_configuration

See Also

AM_mcmc_fit

AM_mcmc_fit *Performs a Gibbs sampling*

Description

The AM_mcmc_fit function performs a Gibbs sampling in order to estimate the mixture comprising the sample data y. The mixture selected must be of a predefined type mix_kernel_hyperparams (defined with AM_mix_hyperparams_* functions, where star * denotes the chosen kernel). Additionally, a prior distribution on the number of mixture components must be specified through mix_components_prior (generated with AM_mix_components_prior_* functions, where * denotes the chosen prior). Similarly, a prior on the weights of the mixture should be specified through mix_weight_prior (defined with AM_mix_weights_prior_* functions). Finally, with mcmc_parameters, the user sets the MCMC parameters for the Gibbs sampler (defined with AM_mcmc_parameters functions).

Usage

```
AM_mcmc_fit(
  y,
  mix_kernel_hyperparams,
  initial_clustering = NULL,
  init_K = NULL,
  fixed_clustering = NULL,
  mix_components_prior = AM_mix_components_prior_pois(),
  mix_weight_prior = AM_mix_weights_prior_gamma(),
  mcmc_parameters = AM_mcmc_parameters()
)
```

Arguments

y	input data, can be a vector or a matrix.
mix_kernel_hyperparams	is a configuration list, defined by *_mix_hyperparams functions, where * denotes the chosen kernel. See AM_mix_hyperparams_multiber, AM_mix_hyperparams_multinorm, AM_mix_hyperparams_uninorm, AM_mix_hyperparams_unipois for more details.
initial_clustering	is a vector CI of initial cluster assignement. If no clustering is specified (either as init_K or init_clustering), then every observation is assigned to its own cluster.
init_K	initial value for the number of cluster. When this is specified, AntMAN initialises the clustering assign usng K-means.
fixed_clustering	if specified, this is the vector CI containing the cluster assignments. This will remain unchanged for every iteration.

`mix_components_prior`
: is a configuration list defined by AM_mix_components_prior_* functions, where * denotes the chosen prior. See `AM_mix_components_prior_dirac`, `AM_mix_components_prior_negbin`, `AM_mix_components_prior_pois` for more details.

`mix_weight_prior`
: is a configuration list defined by AM_weight_prior_* functions, where * denotes the chosen prior specification. See `AM_mix_weights_prior_gamma` for more details.

`mcmc_parameters`
: is a configuration list defined by AM_mcmc_parameters. See `AM_mcmc_parameters` for more details.

Details

If no initial clustering is specified (either as `init_K` or `init_clustering`), then every observation is allocated to a different cluster. If `init_K` is specified then AntMAN initialises the clustering through K-means.

Warning: if the user does not specify init_K or initial_cluster, the first steps can be be time-consuming because of default setting of the initial clustering.

Value

The return value is an `AM_mcmc_output` object.

Examples

```
AM_mcmc_fit( AM_sample_unipois()$y,
             AM_mix_hyperparams_unipois (alpha0=2, beta0=0.2),
           mcmc_parameters = AM_mcmc_parameters(niter=50, burnin=0, thin=1, verbose=0))
```

AM_mcmc_output *S3 class AM_mcmc_output*

Description

Output type of return values from `AM_mcmc_fit`.

Value

`AM_mcmc_output`

See Also

`AM_mcmc_fit`

AM_mcmc_parameters *MCMC Parameters*

Description

This function generates an MCMC parameters list to be used as mcmc_parameters argument within AM_mcmc_fit.

Usage

```
AM_mcmc_parameters(
  niter = 5000,
  burnin = 2500,
  thin = 1,
  verbose = 1,
  output = c("CI", "K"),
  parallel = TRUE,
  output_dir = NULL
)
```

Arguments

niter	Total number of MCMC iterations to be carried out.
burnin	Number of iterations to be considered as burn-in. Samples from this burn-in period are discarded.
thin	Thinning rate. This argument specifies how often a draw from the posterior distribution is stored after burnin, i.e. one every -th samples is saved. Therefore, the toral number of MCMC samples saved is (niter -burnin)/thin. If thin =1, then AntMAN stores every iteration.
verbose	A value from 0 to 4, that specifies the desired level of verbosity (0:None, 1:Warnings, 2:Debug, 3:Extras).
output	A list of parameters output to return.
parallel	Some of the algorithms can be run in parallel using OpenMP. When set to True, this parameter triggers the parallelism.
output_dir	Path to an output dir, where to store all the outputs.

Value

An AM_mcmc_configuration Object. This is a list to be used as mcmc_parameters argument with AM_mcmc_fit.

Examples

```
AM_mcmc_parameters (niter=1000, burnin=10000, thin=50)
AM_mcmc_parameters (niter=1000, burnin=10000, thin=50, output=c("CI","W","TAU"))
```

AM_mcmc_refit *Performs a Gibbs sampling reusing previous configuration*

Description

Similar to AM_mcmc_fit, the AM_mcmc_refit function performs a Gibbs sampling in order to estimate a mixture. However parameters will be reused from a previous result from AM_mcmc_fit.

Usage

```
AM_mcmc_refit(y, fit, fixed_clustering, mcmc_parameters = AM_mcmc_parameters())
```

Arguments

y	input data, can be a vector or a matrix.
fit	previous output from AM_mcmc_fit that is used to setup kernel and priors.
fixed_clustering	is a vector CI of cluster assignment that will remain unchanged for every iterations.
mcmc_parameters	is a configuration list defined by AM_mcmc_parameters.

Details

In practice this function will call AM_mcmc_fit(y, fixed_clustering = fixed_clustering, ...); with the same parameters as previously specified.

Value

The return value is an AM_mcmc_output object.

Examples

```
y = AM_sample_unipois()$y
fit = AM_mcmc_fit( y ,
             AM_mix_hyperparams_unipois (alpha0=2, beta0=0.2),
           mcmc_parameters = AM_mcmc_parameters(niter=20, burnin=0, thin=1, verbose=0))
eam = AM_coclustering(fit)
cluster = AM_salso(eam, "binder")
refit = AM_mcmc_refit(y , fit, cluster,
        mcmc_parameters = AM_mcmc_parameters(niter=20, burnin=0, thin=1, verbose=0));
```

AM_mix_components_prior *S3 class AM_mix_components_prior*

Description

Object returned by AM_mix_components_prior_*.

Value

AM_mix_components_prior

See Also

AM_mix_components_prior_dirac, AM_mix_components_prior_negbin, AM_mix_components_prior_pois

---

AM_mix_components_prior_dirac *Generate a configuration object that contains a Point mass prior*

Description

Generate a configuration object that assigns a Point mass prior to the number of mixture components. This is the simplest option and it requires users to specify a value M^* such that $Pr(M = M^* = 1$.

Usage

```
AM_mix_components_prior_dirac(Mstar)
```

Arguments

Mstar	Fixed value M^* for the number of components.

Value

An AM_mix_components_prior object. This is a configuration list to be used as mix_components_prior argument for AM_mcmc_fit.

See Also

AM_mcmc_fit

Examples

```
AM_mix_components_prior_dirac (Mstar=3)
```

`AM_mix_components_prior_negbin` *Generate a configuration object for a Shifted Negative Binomial prior on the number of mixture components*

Description

This generates a configuration object for a Shifted Negative Binomial prior on the number of mixture components such that

$$q_M(m) = Pr(M = m) = \frac{\Gamma(r + m - 1)}{(m - 1)!\Gamma(r)} p^{m-1}(1 - p)^r, \quad m = 1, 2, 3, \ldots$$

The hyperparameters $p \in (0, 1)$ (probability of success) and $r > 0$ (size) can either be fixed using `r` and `p` or assigned appropriate prior distributions. In the latter case, we assume $p \sim Beta(a_P, b_P)$ and $r \sim Gamma(a_R, b_R)$. In AntMAN we assume the following parametrization of the Gamma density:

$$p(x \mid a, b) = \frac{b^a x^{a-1}}{\Gamma(a)} \exp\{-bx\}, \quad x > 0.$$

Usage

```
AM_mix_components_prior_negbin(
  a_R = NULL,
  b_R = NULL,
  a_P = NULL,
  b_P = NULL,
  R = NULL,
  P = NULL,
  init_R = NULL,
  init_P = NULL
)
```

Arguments

`a_R`	The shape parameter a of the $Gamma(a, b)$ prior distribution for r.
`b_R`	The rate parameter b of the $Gamma(a, b)$ prior distribution for r.
`a_P`	The parameter a of the $Beta(a, b)$ prior distribution for p.
`b_P`	The parameter b of the $Beta(a, b)$ prior distribution for p.
`R`	It allows to fix r to a specific value.
`P`	It allows to fix p to a specific value.
`init_R`	The initial value of r, when specifying `a_R` and `b_R`.
`init_P`	The inivial value of p, when specifying `a_P` and `b_P`.

Details

If no arguments are provided, the default is $r = 1, a_P = 1, b_P = 1$.

Additionally, when init_R and init_P are not specified, there are default values: $init_R = 1$ and $init_P = 0.5$.

Value

An AM_mix_components_prior object. This is a configuration list to be used as mix_components_prior argument for AM_mcmc_fit.

See Also

AM_mcmc_fit

Examples

```
AM_mix_components_prior_negbin (R=1, P=1)
AM_mix_components_prior_negbin ()
```

AM_mix_components_prior_pois

Generate a configuration object for a Poisson prior on the number of mixture components

---

Description

This function generates a configuration object for a Shifted Poisson prior on the number of mixture components such that

$$q_M(m) = Pr(M = m) = \frac{e^{-\Lambda}\Lambda^{m-1}}{(m-1)!}, \quad m = 1, 2, 3, \ldots$$

The hyperparameter Λ can either be fixed using Lambda or assigned a $Gamma(a, b)$ prior distribution with a and b. In AntMAN we assume the following parametrization of the Gamma density:

$$p(x \mid a, b) = \frac{b^a x^{a-1}}{\Gamma(a)} \exp\{-bx\}, \quad x > 0.$$

Usage

```
AM_mix_components_prior_pois(a = NULL, b = NULL, Lambda = NULL, init = NULL)
```

Arguments

a	The shape parameter a of the $Gamma(a, b)$ prior distribution.
b	The rate parameter b of the $Gamma(a, b)$ prior distribution.
Lambda	It allows to set the hyperparameter Λ to be assigned a fixed value.
init	The initial value for Λ, when specifying a and b.

Details

If no arguments are provided, the default is a prior distribution with a = 1 and b = 1.

Value

An AM_mix_components_prior object. This is a configuration list to be used as mix_components_prior argument for AM_mcmc_fit.

See Also

AM_mcmc_fit

Examples

```
components_prior = AM_mix_components_prior_pois (init=3,  a=1, b=1)
```

AM_mix_hyperparams *S3 class AM_mix_hyperparams*

Description

Object type returned by AM_mix_hyperparams_* commands.

Value

AM_mix_hyperparams

See Also

AM_mix_hyperparams_unipois, AM_mix_hyperparams_uninorm, AM_mix_hyperparams_multiber, AM_mix_hyperparams_multinorm

---

AM_mix_hyperparams_multiber *multivariate Bernoulli mixture hyperparameters (Latent Class Analysis)*

Description

Generate a configuration object that defines the prior hyperparameters for a mixture of multivariate Bernoulli. If the dimension of the data is P, then the prior is a product of P independent Beta distributions, Beta(a_{0i}, b_{0i}). Therefore, the vectors of hyperparameters, a0 and b0, are P-dimensional. Default is (a0= c(1,....,1),b0= c(1,....,1)).

Usage

```
AM_mix_hyperparams_multiber(a0, b0)
```

Arguments

a0	The a0 hyperparameters.
b0	The b0 hyperparameters.

Value

An AM_mix_hyperparams object. This is a configuration list to be used as mix_kernel_hyperparams argument for AM_mcmc_fit.

Examples

```
AM_mix_hyperparams_multiber (a0= c(1,1,1,1),b0= c(1,1,1,1))
```

AM_mix_hyperparams_multinorm

multivariate Normal mixture hyperparameters

Description

Generate a configuration object that specifies a multivariate Normal mixture kernel, where users can specify the hyperparameters for the conjugate prior of the multivariate Normal mixture. We assume that the data are d-dimensional vectors $\boldsymbol{y}_i$, where $\boldsymbol{y}_i$ are i.i.d Normal random variables with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. The conjugate prior is

$$\pi(\boldsymbol{\mu}, \boldsymbol{\Sigma} \mid \boldsymbol{m}_0, \kappa_0, \nu_0, \boldsymbol{\Lambda}_0) = \pi_\mu(\boldsymbol{\mu}|\boldsymbol{\Sigma}, \boldsymbol{m}_0, \kappa_0)\pi_\Sigma(\boldsymbol{\Sigma} \mid \nu_0, \boldsymbol{\Lambda}_0),$$

$$\pi_\mu(\boldsymbol{\mu}|\boldsymbol{\Sigma}, \boldsymbol{m}_0, \kappa_0) = \frac{\sqrt{\kappa_0^d}}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}} \exp\left(-\frac{\kappa_0}{2}(\boldsymbol{\mu} - \boldsymbol{m}_0)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \boldsymbol{m}_0)\right), \qquad \boldsymbol{\mu} \in \mathcal{R}^d,$$

$$\pi_\Sigma(\boldsymbol{\Sigma} \mid \nu_0, \boldsymbol{\Lambda}_0) = \frac{|\boldsymbol{\Lambda}_0|^{\nu_0/2}}{2^{\nu_0 d/2}\Gamma_d(\frac{\nu_0}{2})} |\boldsymbol{\Sigma}|^{-(\nu_0+d+1)/2} e^{-\frac{1}{2}\mathrm{tr}(\boldsymbol{\Lambda}_0\boldsymbol{\Sigma}^{-1})}, \qquad \boldsymbol{\Sigma}^2 > 0,$$

where mu0 corresponds to $\boldsymbol{m}_0$, ka0 corresponds to κ_0, nu0 to ν_0, and Lam0 to Λ_0.

Usage

```
AM_mix_hyperparams_multinorm(mu0 = NULL, ka0 = NULL, nu0 = NULL, Lam0 = NULL)
```

Arguments

mu0	The hyperparameter $\boldsymbol{m}_0$.
ka0	The hyperparameter κ_0.
nu0	The hyperparameter ν_0.
Lam0	The hyperparameter Λ_0.

Details

Default is `(mu0=c(0,..,0), ka0=1, nu0=Dim+2, Lam0=diag(Dim))` with `Dim` is the dimension of the data `y`. We advise the user to set ν_0 equal to at least the dimension of the data, `Dim`, plus 2.

Value

An `AM_mix_hyperparams` object. This is a configuration list to be used as `mix_kernel_hyperparams` argument for `AM_mcmc_fit`.

Examples

```
AM_mix_hyperparams_multinorm ()
```

`AM_mix_hyperparams_uninorm`

univariate Normal mixture hyperparameters

Description

Generate a configuration object that specifies a univariate Normal mixture kernel, where users can specify the hyperparameters of the Normal-InverseGamma conjugate prior. As such, the kernel is a Gaussian distribution with mean μ and variance σ^2. The prior on (μ, σ^2) the Normal-InverseGamma:

$$\pi(\mu, \sigma^2 \mid m_0, \kappa_0, \nu_0, \sigma_0^2) = \pi_\mu(\mu|\sigma^2, m_0, \kappa_0)\pi_{\sigma^2}(\sigma^2 \mid \nu_0, \sigma_0^2),$$

$$\pi_\mu(\mu|\sigma^2, m_0, \kappa_0) = \frac{\sqrt{\kappa_0}}{\sqrt{2\pi\sigma^2},} \exp^{-\frac{\kappa_0}{2\sigma^2}(\mu - m_0)^2}, \qquad \mu \in \mathcal{R},$$

$$\pi_{\sigma^2}(\sigma^2 \mid \nu_0, \sigma_0^2) = \frac{\sigma_0^{2\nu_0}}{\Gamma(\nu_0)}(1/\sigma^2)^{\nu_0+1} \exp\left(-\frac{\sigma_0^2}{\sigma^2}\right), \qquad \sigma^2 > 0.$$

Usage

```
AM_mix_hyperparams_uninorm(m0, k0, nu0, sig02)
```

Arguments

`m0`	The m_0 hyperparameter.
`k0`	The κ_0 hyperparameter.
`nu0`	The ν_0 hyperparameter.
`sig02`	The σ_0^2 hyperparameter.

Details

m_0 corresponds `m0`, κ_0 corresponds `k0`, ν_0 corresponds `nu0`, and σ_0^2 corresponds `sig02`.

If hyperparameters are not specified, the default is `m0=0`, `k0=1`, `nu0=3`, `sig02=1`.

Value

An AM_mix_hyperparams object. This is a configuration list to be used as mix_kernel_hyperparams argument for AM_mcmc_fit.

Examples

```
#### This example ...

data(galaxy)
y_uvn = galaxy
mixture_uvn_params = AM_mix_hyperparams_uninorm  (m0=20.83146, k0=0.3333333,
                                                  nu0=4.222222, sig02=3.661027)

mcmc_params        = AM_mcmc_parameters(niter=2000, burnin=500, thin=10, verbose=0)
components_prior   = AM_mix_components_prior_pois (init=3,  a=1, b=1)
weights_prior      = AM_mix_weights_prior_gamma(init=2, a=1, b=1)

fit <- AM_mcmc_fit(
  y = y_uvn,
  mix_kernel_hyperparams = mixture_uvn_params,
  mix_components_prior =components_prior,
  mix_weight_prior = weights_prior,
  mcmc_parameters = mcmc_params)

summary (fit)
plot (fit)
```

AM_mix_hyperparams_unipois

univariate Poisson mixture hyperparameters

Description

Generate a configuration object that specifies a univariate Poisson mixture kernel, where users can specify the hyperparameters of the conjugate Gamma prior, i.e. the kernel is a $Poisson(\tau)$ and $\tau \sim Gamma(\alpha_0, \beta_0)$. In AntMAN we assume the following parametrization of the Gamma density:

$$p(x \mid a, b) = \frac{b^a x^{a-1}}{\Gamma(a)} \exp\{-bx\}, \quad x > 0.$$

Usage

```
AM_mix_hyperparams_unipois(alpha0, beta0)
```

Arguments

alpha0	The shape hyperparameter α_0.
beta0	The rate hyperparameter β_0.

Details

Note that by default, alpha0=1 and beta0=1.

Value

An AM_mix_hyperparams object. This is a configuration list to be used as mix_kernel_hyperparams argument for AM_mcmc_fit.

Examples

```
AM_mix_hyperparams_unipois (alpha0=2, beta0=0.2)
```

AM_mix_weights_prior *S3 class AM_mix_weights_prior*

Description

Object type returned by AM_mix_weights_prior_* commands.

Value

AM_mix_weights_prior

See Also

AM_mix_weights_prior_gamma

AM_mix_weights_prior_gamma *specify a prior on the hyperparameter γ for the Dirichlet mixture weights prior*

Description

Generate a configuration object to specify a prior on the hyperparameter γ for the Dirichlet prior on the mixture weights. We assume $\gamma \sim Gamma(a, b)$. Alternatively, we can fix γ to a specific value. Default is $\gamma = 1/N$, where N is the number of observations. In AntMAN we assume the following parametrization of the Gamma density:

$$p(x \mid a, b) = \frac{b^a x^{a-1}}{\Gamma(a)} \exp\{-bx\}, \quad x > 0.$$

Usage

```
AM_mix_weights_prior_gamma(a = NULL, b = NULL, gamma = NULL, init = NULL)
```

Arguments

a	The shape parameter a of the Gamma prior.
b	The rate parameter b of the Gamma prior.
gamma	It allows to fix γ to a specific value.
init	The init value for γ, when we assume γ random.

Value

A AM_mix_weights_prior object. This is a configuration list to be used as mix_weight_prior argument for AM_mcmc_fit.

Examples

```
AM_mix_weights_prior_gamma (a=1, b=1)
AM_mix_weights_prior_gamma (a=1, b=1, init=1)
AM_mix_weights_prior_gamma (gamma = 3)
AM_mix_weights_prior_gamma ()
```

AM_plot_chaincor *Plot the Autocorrelation function*

Description

Given an AM_mcmc_output object, this function produces the autocorrelation function bars describing the MCMC results. AM_plot_chaincor makes use of bayesplot's plotting function mcmc_acf_bar (Gabry et al. 2019).

Usage

```
AM_plot_chaincor(x, tags = NULL, lags = NULL, title = "MCMC Results")
```

Arguments

x	An AM_mcmc_output object, produced by calling AM_mcmc_fit.
tags	A list of variables to consider. This function only produces meaningful plots for variables that have fixed dimension across the draws. If not specified, plots pertaining to M and K will be produced. This function is built upon bayesplot's mcmc_acf_bar.
lags	An integer specifying the number of lags to plot. If no value is specified, the default number of lags shown is half the total number of iterations.
title	Title for the plot.

Value

A ggplot object.

AM_plot_density *Plot the density of variables from* AM_mcmc_output *object*

Description

Given an AM_mcmc_output object, AM_plot_density plots the posterior density of the specified variables of interest. AM_plot_density makes use of bayesplot's plotting function mcmc_areas (Gabry et al. 2019).

Usage

```
AM_plot_density(x, tags = NULL, title = "MCMC Results")
```

Arguments

x	An AM_mcmc_output fit object, produced by calling AM_mcmc_fit.
tags	A list of variables to consider. This function only produces meaningful plots for variables that have fixed dimension across the draws.
title	Title for the plot.

Value

a ggplot object visualising the posterior density of the specified variables.

AM_plot_mvb_cluster_frequency

Visualise the cluster frequency plot for the multivariate bernoulli model

Description

Given an AM_mcmc_output object, and the data the model was fit on, this function will produce a cluster frequency plot for the multivariate bernoulli model.

Usage

```
AM_plot_mvb_cluster_frequency(
  fit,
  y,
  x_lim_param = c(0.8, 7.2),
  y_lim_param = c(0, 1)
)
```

Arguments

fit	An AM_mcmc_output fit object, produced by calling AM_mcmc_fit.
y	A matrix containing the y observations which produced fit.
x_lim_param	A vector with two elements describing the plot's x_axis scale, e.g. c(0.8, 7.2).
y_lim_param	A vector with two elements describing the plot's y_axis scale, e.g. c(0, 1).

Value

No return value. Called for side effects.

AM_plot_pairs *Plot* AM_mcmc_output *scatterplot matrix*

Description

visualise a matrix of plots describing the MCMC results. This function is built upon GGally's plotting function ggpairs (Schloerke et al. 2021).

Usage

```
AM_plot_pairs(x, tags = NULL, title = "MCMC Results")
```

Arguments

x	an AM_mcmc_output object, produced by calling AM_mcmc_fit.
tags	A list of variables to consider for plotting. This function only produces meaningful plots for variables that have fixed dimension across the draws. If not specified, plots pertaining to M and K will be produced.
title	Title for the plot.

Value

Same as ggpairs function, a ggmatrix object that if called, will print.

AM_plot_pmf *Plot the probability mass function of variables from* AM_mcmc_output *object*

Description

Given an AM_mcmc_output object, AM_plot_pmf plots the posterior probability mass function of the specified variables.

Usage

```
AM_plot_pmf(x, tags = NULL, title = "MCMC Results")
```

Arguments

x	An AM_mcmc_output object, produced by calling AM_mcmc_fit.
tags	A list of variables to consider. If not specified, the pmf of both M and K will be plotted.
title	Title for the plot.

Value

No return value. Called for side effects.

AM_plot_similarity_matrix *Plot the Similarity Matrix*

Description

Given an AM_mcmc_output object, this function will produce an image of the Similarity Matrix.

Usage

```
AM_plot_similarity_matrix(x, loss, ...)
```

Arguments

x	An AM_mcmc_output fit object, produced by calling AM_mcmc_fit.
loss	Loss function to minimise. Specify either "VI" or "binder". If not specified, the default loss to minimise is "binder".
...	All additional parameters wil lbe pass to the image command.

Value

No return value. Called for side effects.

AM_plot_traces *Plot traces of variables from an AM_mcmc_output object*

Description

Given an AM_mcmc_output object, AM_plot_traces visualises the traceplots of the specified variables involved in the MCMC inference. AM_plot_traces is built upon bayesplot's mcmc_trace (Gabry et al. 2019).

Usage

```
AM_plot_traces(x, tags = NULL, title = "MCMC Results")
```

Arguments

x	An AM_mcmc_output fit object, produced by calling AM_mcmc_fit.
tags	A list of variables to consider. This function only produces meaningful plots for variables that have fixed dimension across the draws. If not specified, plots pertaining to M and K will be produced.
title	Title for the plot

Value

No return value. Called for side effects.

AM_plot_values *Plot posterior interval estimates obtained from MCMC draws*

Description

Given an object of class AM_mcmc_fit, AM_plot_values visualises the interval estimates of the specified variables involved in the MCMC inference. AM_plot_values is built upon bayesplot's mcmc_intervals (Gabry et al. 2019).

Usage

```
AM_plot_values(x, tags = NULL, title = "MCMC Results")
```

Arguments

x	An AM_mcmc_output fit object, produced by calling AM_mcmc_fit.
tags	A list of variables to consider. This function only produces meaningful plots for variables that have fixed dimension across the draws. If not specified, plots pertaining to M and K will be produced.
title	Title for the plot.

Value

No return value. Called for side effects.

`AM_prior` *S3 class AM_prior*

Description

Object type returned by `AM_prior_*` commands.

Value

`AM_prior`

See Also

`AM_prior_K_Delta`, `AM_prior_K_Pois`, `AM_prior_K_NegBin`

`AM_prior_K_Delta` *Computes the prior on the number of clusters*

Description

This function computes the prior on the number of clusters, i.e. occupied components of the mixture for a Finite Dirichlet process when the prior on the component-weights of the mixture is a Dirichlet with parameter `gamma` (i.e. when unnormalised weights are distributed as Gamma(γ,1)). This function can be used when the number of components is fixed to M^*, i.e. a Dirac prior assigning mass only to M^* is assumed. See (Argiento and Iorio 2019) There are no default values.

Usage

```
AM_prior_K_Delta(n, gamma, Mstar)
```

Arguments

`n`	The sample size.
`gamma`	The `gamma` parameter of the Dirichlet distribution.
`Mstar`	The number of component of the mixture.

Value

an `AM_prior` object, that is a vector of length n, reporting the values `V(n,k)` for `k=1,...,n`.

Examples

```
n <- 82
gam_de <- 0.1743555
Mstar <- 12
prior_K_de <- AM_prior_K_Delta(n,gam_de, Mstar)
plot(prior_K_de)
```

AM_prior_K_NegBin *computes the prior number of clusters*

Description

This function computes the prior on the number of clusters, i.e. occupied component of the mixture for a Finite Dirichlet process when the prior on the component-weights of the mixture is a Dirichlet with parameter gamma (i.e. when unnormalized weights are distributed as Gamma(γ,1)). This function can be used when the prior on the number of components is Negative Binomial with parameter $r > 0$ and $0 < p < 1$, with mean $mu = 1 + r * p/(1 - p)$. See (Argiento and Iorio 2019) for more details.

Usage

```
AM_prior_K_NegBin(n, gamma, r, p)
```

Arguments

n	The sample size.
gamma	The gamma parameter of the Dirichlet distribution.
r	The dispersion parameter r of the Negative Binomial.
p	The probability of failure parameter p of the Negative Binomial.

Details

There are no default values.

Value

an AM_prior object, that is a vector of length n, reporting the values V(n,k) for k=1,...,n.

Examples

```
n <- 50
gamma <- 1
r <- 0.1
p <- 0.91
gam_nb <- 0.2381641
prior_K_nb <-  AM_prior_K_NegBin(n,gam_nb,r,p)
plot(prior_K_nb)
```

`AM_prior_K_Pois` *Computes the prior number of clusters*

Description

This function computes the prior on the number of clusters, i.e. occupied components of the mixture for a Finite Dirichlet process when the prior on the component-weights of the mixture is a Dirichlet with parameter `gamma` (i.e. when unnormalized weights are distributed as Gamma(γ,1)). This function can be used when the prior on the number of components is Shifted Poisson of parameter `Lambda`. See (Argiento and Iorio 2019) for more details.

Usage

```
AM_prior_K_Pois(n, gamma, Lambda)
```

Arguments

`n`	The sample size.
`gamma`	The `gamma` parameter of the Dirichlet distribution.
`Lambda`	The `Lambda` parameter of the Poisson.

Details

There are no default values.

Value

an `AM_prior` object, that is a vector of length `n`, reporting the values of the prior on the number of clusters induced by the prior on `M` and `w`, i.e. `p^*_k` for `k=1,...,n`. See (Argiento and Iorio 2019) for more details.

Examples

```
n <- 82
Lambda <- 10
gam_po <- 0.1550195
prior_K_po <-  AM_prior_K_Pois(n,gam_po,Lambda)
plot(prior_K_po)
```

`AM_salso` *Sequentially Allocated Latent Structure Optimisation*

Description

Heuristic partitioning to minimise the expected loss function with respect to a given expected adjacency matrix. This function is built upon R-package salso's implementation of the `salso` function. See salso (Dahl et al. 2021) for more details.

Usage

```
AM_salso(
  eam,
  loss,
  maxNClusters = 0,
  nRuns = 16,
  maxZealousAttempts = 10,
  probSequentialAllocation = 0.5,
  nCores = 0
)
```

Arguments

`eam`	a co-clustering/ clustering matrix. See salso for more information on which matrix to use.
`loss`	the recommended loss functions to be used are the "binder" or "VI". However, other loss functions that are supported can be found in the R-package salso's salso function.
`maxNClusters`	Maximum number of clusters to be considered. The actual number of clusters searched may be lower. Default is 0.
`nRuns`	Number of runs to try.
`maxZealousAttempts`	Maximum number of tries for zealous updates. See salso for more information.
`probSequentialAllocation`	The probability of using sequential allocation instead of random sampling via sample(1:K,ncol(x),TRUE), where K is maxNClusters. Default is 0.5. See salso for more information. argument.
`nCores`	Number of CPU cores to engage. Default is 0.

Value

A numeric vector describing the estimated partition. The integer values represent the cluster labels of each item respectively.

Source

David B. Dahl and Devin J. Johnson and Peter Müller (2021). salso: Search Algorithms and Loss Functions for Bayesian Clustering. R package version 0.2.15.

`AntMAN` *AntMAN: A package for fitting finite Bayesian Mixture models with a random number of components*

Description

AntMAN: Anthology of Mixture ANalysis tools AntMan is an R package fitting Finite Bayesian Mixture models with a random number of components. The MCMC algorithm behind AntMAN is based on point processes and offers a more computationally efficient alternative to the Reversible Jump. Different mixture kernels can be specified: univariate Gaussian, multivariate Gaussian, univariate Poisson, and multivariate Bernoulli (Latent Class Analysis). For the parameters characterising the mixture kernel, we specify conjugate priors, with possibly user specified hyper-parameters. We allow for different choices on the prior on the number of components: Shifted Poisson, Negative Binomial, and Point Masses (i.e. mixtures with fixed number of components).

Package Philosophy

The main function of the AntMAN package is `AM_mcmc_fit`. AntMAN performs a Gibbs sampling in order to fit, in a Bayesian framework, a mixture model of a predefined type `mix_kernel_hyperparams` given a sample `y`. Additionally AntMAN allows the user to specify a prior on the number of components `mix_components_prior` and on the weights `mix_weight_prior` of the mixture. MCMC parameters `mcmc_parameters` need to be given as argument for the Gibbs sampler (number of interations, burn-in, ...). Initial values for the number of clusters (`init_K`) or a specific clustering allocation (`init_clustering`) can also be user-specified. Otherwise, by default, we initialise each element of the sample `y` to a different cluster allocation. This choice can be computationally inefficient.

For example, in order to identify clusters over a population of patients given a set of medical assumptions:

```
mcmc = AM_mcmc_parameters(niter=20000)
mix = AM_mix_hyperparams_multiber ()
fit = AM_mcmc_fit (mix, mcmc)
summary (fit)
```

In this example `AM_mix_hyperparams_multiber` is one of the possible mixtures to use.

AntMAN currently support four different mixtures :

```
AM_mix_hyperparams_unipois(alpha0, beta0)
AM_mix_hyperparams_uninorm(m0, k0, nu0, sig02)
AM_mix_hyperparams_multiber(a0, b0)
AM_mix_hyperparams_multinorm(mu0, ka0, nu0, Lam0)
```

Additionally, three types of kernels on the prior number of components are available:

```
AM_mix_components_prior_pois()
AM_mix_components_prior_negbin()
AM_mix_components_prior_dirac()
```

For example, in the context of image segmentation, if we know that there are 10 colours present, a prior dirac can be used :

```
mcmc = AM_mcmc_parameters(niter=20000)
mix = AM_mix_hyperparams_multinorm ()
prior_component = AM_mix_components_prior_dirac(10) #  10 colours present
fit = AM_mcmc_fit (mix, prior_component, mcmc)
summary (fit)
```

`brain` *Teen Brain Images from the National Institutes of Health, U.S.*

Description

Picture of brain activities from a teenager consuming drugs.

Usage

```
brain
```

Format

A list that contains `dim` a (W:width,H:height) pair, and `pic` a data frame (W*H pixels image in RGB format).

Source

https://www.flickr.com/photos/nida-nih/29741916012

References

Crowley TJ, Dalwani MS, Mikulich-Gilbertson SK, Young SE, Sakai JT, Raymond KM, et al. (2015) Adolescents' Neural Processing of Risky Decisions: Effects of Sex and Behavioral Disinhibition. PLoS ONE 10(7): e0132322. doi:10.1371/journal.pone.0132322

Examples

```
data(brain)
```

carcinoma *Carcinoma dataset*

Description

The carcinoma data from Agresti (2002, 542) consist of seven dichotomous variables representing the ratings by seven pathologists of 118 slides on the presence or absence of carcinoma. The purpose of studying this data is to model "interobserver agreement" by examining how subjects might be divided into groups depending upon the consistency of their diagnoses.

Usage

```
carcinoma
```

Format

A data frame with 118 rows and 7 variables (from A to G).

References

Agresti A (2002). Categorical Data Analysis. John Wiley & Sons, Hoboken.

Examples

```
data(carcinoma)
```

galaxy *Galaxy velocities dataset*

Description

This data set considers the physical information of velocities (10^3 km/second) for 82 galaxies reported by Roeder (1990). These are drawn from six well-separated conic sections of the Corona Borealis region.

Usage

```
galaxy
```

Format

A data frame with X rows and Y variables.

A numeric vector giving the speed of galaxies (1000*(km/second))

Source

Roeder, K. (1990). Density estimation with confidence sets exemplified by superclusters and voids in the galaxies, Journal of the American Statistical Association, 85: 617-624.

Examples

```
data(galaxy)
```

`plot.AM_mcmc_output` *plot AM_mcmc_output*

Description

Given an `AM_mcmc_output` object, this function plots some useful information about the MCMC results regarding M and K. Besides the PMFs, some of the diagnostic plots of the MCMC chain are visualised.

Usage

```
## S3 method for class 'AM_mcmc_output'
plot(x, ...)
```

Arguments

`x`	an `AM_mcmc_output` object.
`...`	all additional parameters are ignored.

Value

NULL. Called for side effects.

`plot.AM_prior` *plot AM_prior*

Description

plot the prior on the number of clusters for a given `AM_prior` object.

Usage

```
## S3 method for class 'AM_prior'
plot(x, ...)
```

Arguments

x	an AM_prior object. See AM_prior_K_Delta, AM_prior_K_NegBin, AM_prior_K_Pois for more details.
...	all additional parameters are ignored.

Value

NULL. Called for side effects.

---

said *Usage frequency of the word "said" in the Brown corpus*

Description

Usage frequency of the word "said" in the Brown corpus

Usage

```
said
```

Format

A list with 500 observations on the frequency of said in different texts.

Source

https://www.kaggle.com/nltkdata/brown-corpus

References

Francis, W., and Kucera, H. (1982) Frequency Analysis of English Usage, Houghton Mifflin Company, Boston.

Examples

```
data(said)
```

summary.AM_mcmc_configuration *summary information of the AM_mcmc_configuration object*

Description

Given an AM_mcmc_configuration object, this function prints the summary information of the specified mcmc configuration.

Usage

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

Arguments

object	an AM_mcmc_configuration object.
...	all additional parameters are ignored

Value

NULL. Called for side effects.

See Also

AM_mcmc_parameters

summary.AM_mcmc_output *summary information of the AM_mcmc_output object*

Description

Given an AM_mcmc_output object, this function prints the summary information pertaining to the given model output.

Usage

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

Arguments

object	a AM_mcmc_output object
...	all additional parameters are ignored

Value

NULL. Called for side effects.

See Also

AM_mcmc_fit, AM_mcmc_refit

---

summary.AM_mix_components_prior
summary information of the AM_mix_components_prior object

Description

Given an AM_mix_components_prior object, this function prints the summary information of the specified prior on the number of components.

Usage

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

Arguments

object	an AM_mix_components_prior object.
...	all additional parameters are ignored.

Value

NULL. Called for side effects.

See Also

AM_mix_components_prior

summary.AM_mix_hyperparams *summary information of the AM_mix_hyperparams object*

Description

Given an AM_mix_hyperparams object, this function prints the summary information of the specified mixture hyperparameters.

Usage

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

Arguments

object	an AM_mix_hyperparams object.
...	all additional parameters are ignored.

Value

NULL. Called for side effects.

See Also

AM_mix_hyperparams

summary.AM_mix_weights_prior *summary information of the AM_mix_weights_prior object*

Description

Given an AM_mix_weights_prior object, this function prints the summary information of the specified mixture weights prior.

Usage

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

Arguments

object	an AM_mix_weights_prior object.
...	all additional parameters are ignored.

Value

NULL. Called for side effects.

See Also

AM_mix_weights_prior

---

summary.AM_prior *summary information of the AM_prior object*

Description

Given an AM_prior object, this function prints the summary information of the specified prior on the number of clusters.

Usage

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

Arguments

object	an AM_prior object. See AM_prior_K_Delta, AM_prior_K_NegBin, AM_prior_K_Pois for more details.
...	all additional parameters are ignored.

Value

NULL. Called for side effects.

See Also

AM_prior

Index