Package 'jDirichletMixtureModels'

April 21, 2018

Title Dirichlet Mixture Models for Clustering using Julia backend

Version 0.0.0.9000

Description This package provides utilities for clustering using Dirichlet Process mixture models using Julia for backend computation.
It supports a number of existing conjugate distribution pairs, as well as user-specified distributions. Currently, the package only
supports clustering using conjugate distributions using the methods in Markov Chain Sampling Methods for Dirichlet Process Mixture
Models by Radford Neal. Clustering using non-conjugate distributions is under active development.
Julia is a high-level, high-performance dynamic programming language for numerical computing. This package is based on the Julia package
DirichletMixtureModels (see https://github.com/krylea/DirichletMixtureModels.jl), which is being co-developed alongside this package.

URL https://github.com/nsdumont/jDirichletMixtureModels

Depends R (>= 3.4.0)

Imports JuliaCall

Suggests data.table, ggplot2, scatterplot3d, testthat

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1.9000

Author Nicole Dumont [aut, cre],
Kira Selby [aut]

Maintainer Nicole Dumont <ns2dumont@uwaterloo.ca>

RemoteType local

RemoteUrl /Users/nicoledumont/Documents/GitHub/jDirichletMixtureModels

RemoteSha NA

RemoteBranch master

RemoteUsername nsdumont

RemoteRepo jDirichletMixtureModels

R topics documented:

`aggregationData` *Aggregation Data*

Description

N=788, k=7, D=2

Usage

```
data(aggregationData, package = "jDirichletMixtureModels")
```

Format

An object of class `matrix` with 788 rows and 2 columns.

Source

Clustering basic benchmark

References

A. Gionis, H. Mannila, and P. Tsaparas, Clustering aggregation. ACM Transactions on Knowledge Discovery from Data (TKDD), 2007. 1(1): p. 1-30.

Examples

```
## Not run:
Xdata <- data(aggregationData, package = "jDirichletMixtureModels")
states <- dmm.cluster(model, Xdata)
dmm.plot(states[[1]]$labeledData)

## End(Not run)
```

`dmm.addfile` *Add a Julia file to be accessible.*

Description

This function must be called before runnning the `dmm.cluster` function using a JModel (see `dmm.JModel`).

Usage

```
dmm.addfile(filename)
```

Arguments

`filename`	The name of the Julia file.

`dmm.BaseModel` *Create a model using bulit-in conjugate models*

Description

Create an model object to be used in the `dmm.cluster` function, using the packages bulit-in conjugate models. Function `dmm.model` is an alternative method.

Usage

```
model <- dmm.BaseModel(typename, params)
```

Arguments

typename	A string. The name of the predefined conjugate prior you wish to use. Options listed under details. "MultivariateNormalModel" is the default.
params	A list of the hyperparameter values for the likelihood functions. Many model have default params values and thus can be made without passing any params. See documentation for what parameters a given model may take.
data	In lieu of explicit hyperparameter values, some models can infer good hyperparameter values from given data. This option is supported for MultivariateNormalModel and UnivariateNormalModel, as well as UnivariateNormalKnownSigma.

Details

Bulit-in models avaible are: "MultivariateNormalModel" (default), "UnivariateNormalModel", "UnivariateNormalKnownSigma", "UnivariateExponentialModel".

Value

A model object of type BaseModel which can be passed to dmm.cluster.

---

dmm.benchmark *To get MCMC computation times*

Description

This function is the same as dmm.cluster except instead of returning the states it returns the time it took to do preprocessing computations, the MCMC computation, and the postprocessing computations. Currently avaible for testing purposes.

Usage

```
dmm.benchmark(model, Xdata, alpha = 1, m_prior = 3, m_post = 3,
  iters = 5000, burnin = 200, shuffled = TRUE)
```

Arguments

model	An object returned by dmm.model().
Xdata	A 1D array of length N (univariate case) or 2D array of size N-by-d (mulitvariate case), where d is the dimensionailty of the data and N is the number of observations.
alpha	A float. The concentration parameter. Default is 1.0.
m_prior	An integer. Optionally paramter only used in non-conjugate case. Default is 3.
m_post	An integer. Optionally paramter only used in non-conjugate case. Default is 3.
iters	An integer. Number of iterations. Default is 5000.
burnin	An integer. Amount of burn-in. Default is 200.
shuffled	A logical. Whether or not to shuffle the data. Default is true.

Details

Performs `iters` iterations of Algorithm 2 (in conjugate case) or Algorithm 8 (in non-conjugate case) from Neal(2000) to generate possible clusters for the data in `Xdata`, using the model in `model`, with concentration parameter `alpha`. In the 1D case, `Xdata` is assumed to be a 1D array of floats. In the 2D case, `Xdata` is assumed to be a dxN array of floats, where the data is d-dimensional and N is the number of datapoints. Returns a dataframe of the time it took to do preprocessing computations, the MCMC computation, and the postprocessing computations.

Value

A dataframe of the time in seconds it took to do preprocessing computations, the MCMC computation, and the postprocessing computations.

`dmm.benchmark.BaseModel`

To get MCMC computation times

Description

This function is the same as `dmm.cluster` except instead of returning the states it returns the time it took to do preprocessing computations, the MCMC computation, and the postprocessing computations. Currently avaible for testing purposes.

Usage

```
## S3 method for class 'BaseModel'
dmm.benchmark(model, Xdata, alpha = 1, iters = 5000,
  burnin = 200, shuffled = TRUE)
```

Arguments

`model`	An object returned by `dmm.model()`.
`Xdata`	A 1D array of length N (univariate case) or 2D array of size N-by-d (mulitvariate case), where d is the dimensionailty of the data and N is the number of observations.
`alpha`	A float. The concentration parameter. Default is 1.0.
`iters`	An integer. Number of iterations. Default is 5000.
`burnin`	An integer. Amount of burn-in. Default is 200.
`shuffled`	A logical. Whether or not to shuffle the data. Default is true.
`m_prior`	An integer. Optionally paramter only used in non-conjugate case. Default is 3.
`m_post`	An integer. Optionally paramter only used in non-conjugate case. Default is 3.

Details

Performs `iters` iterations of Algorithm 2 (in conjugate case) or Algorithm 8 (in non-conjugate case) from Neal(2000) to generate possible clusters for the data in `Xdata`, using the model in `model`, with concentration parameter `alpha`. In the 1D case, `Xdata` is assumed to be a 1D array of floats. In the 2D case, `Xdata` is assumed to be a dxN array of floats, where the data is d-dimensional and N is the number of datapoints. Returns a dataframe of the time it took to do preprocessing computations, the MCMC computation, and the postprocessing computations.

Value

A dataframe of the time in seconds it took to do preprocessing computations, the MCMC computation, and the postprocessing computations.

`dmm.benchmark.JConjugateModel`

To get MCMC computation times

Description

This function is the same as `dmm.cluster` except instead of returning the states it returns the time it took to do preprocessing computations, the MCMC computation, and the postprocessing computations. Currently avaible for testing purposes.

Usage

```
## S3 method for class 'JConjugateModel'
dmm.benchmark(model, Xdata, alpha = 1,
  m_prior = 3, m_post = 3, iters = 5000, burnin = 200,
  shuffled = TRUE)
```

Arguments

`model`	An object returned by `dmm.model()`.
`Xdata`	A 1D array of length N (univariate case) or 2D array of size N-by-d (mulitvariate case), where d is the dimensionailty of the data and N is the number of observations.
`alpha`	A float. The concentration parameter. Default is 1.0.
`m_prior`	An integer. Optionally paramter only used in non-conjugate case. Default is 3.
`m_post`	An integer. Optionally paramter only used in non-conjugate case. Default is 3.
`iters`	An integer. Number of iterations. Default is 5000.
`burnin`	An integer. Amount of burn-in. Default is 200.
`shuffled`	A logical. Whether or not to shuffle the data. Default is true.

Details

Performs `iters` iterations of Algorithm 2 (in conjugate case) or Algorithm 8 (in non-conjugate case) from Neal(2000) to generate possible clusters for the data in `Xdata`, using the model in `model`, with concentration parameter `alpha`. In the 1D case, `Xdata` is assumed to be a 1D array of floats. In the 2D case, `Xdata` is assumed to be a dxN array of floats, where the data is d-dimensional and N is the number of datapoints. Returns a dataframe of the time it took to do preprocessing computations, the MCMC computation, and the postprocessing computations.

Value

A dataframe of the time in seconds it took to do preprocessing computations, the MCMC computation, and the postprocessing computations.

`dmm.benchmark.JNonConjugateModel` *To get MCMC computation times*

Description

This function is the same as `dmm.cluster` except instead of returning the states it returns the time it took to do preprocessing computations, the MCMC computation, and the postprocessing computations. Currently avaible for testing purposes.

Usage

```
## S3 method for class 'JNonConjugateModel'
dmm.benchmark(model, Xdata, alpha = 1,
  m_prior = 3, m_post = 4, iters = 5000, burnin = 200,
  shuffled = TRUE)
```

Arguments

`model`	An object returned by `dmm.model()`.
`Xdata`	A 1D array of length N (univariate case) or 2D array of size N-by-d (mulitvariate case), where d is the dimensionailty of the data and N is the number of observations.
`alpha`	A float. The concentration parameter. Default is 1.0.
`m_prior`	An integer. Optionally paramter only used in non-conjugate case. Default is 3.
`m_post`	An integer. Optionally paramter only used in non-conjugate case. Default is 3.
`iters`	An integer. Number of iterations. Default is 5000.
`burnin`	An integer. Amount of burn-in. Default is 200.
`shuffled`	A logical. Whether or not to shuffle the data. Default is true.

Details

Performs `iters` iterations of Algorithm 2 (in conjugate case) or Algorithm 8 (in non-conjugate case) from Neal(2000) to generate possible clusters for the data in `Xdata`, using the model in `model`, with concentration parameter `alpha`. In the 1D case, `Xdata` is assumed to be a 1D array of floats. In the 2D case, `Xdata` is assumed to be a dxN array of floats, where the data is d-dimensional and N is the number of datapoints. Returns a dataframe of the time it took to do preprocessing computations, the MCMC computation, and the postprocessing computations.

Value

A dataframe of the time in seconds it took to do preprocessing computations, the MCMC computation, and the postprocessing computations.

`dmm.cluster` *Use a Dirichlet Mixture Model on data to get cluster labels and cluster parameter values.*

Description

Use a Dirichlet Mixture Model on data to get cluster labels and cluster parameter values.

Usage

```
dmm.cluster(model, Xdata, alpha=1.0, m_prior=3, m_post=3, iters=5000, burnin=200, shuffled=TRUE)
```

Arguments

`model`	An object returned by `dmm.model()`.
`Xdata`	A 1D array of length N (univariate case) or 2D array of size N-by-d (mulit-variate case), where d is the dimensionailty of the data and N is the number of observations.
`alpha`	A float. The concentration parameter. Default is 1.0.
`m_prior`	An integer. Optionally paramter only used in non-conjugate case. Default is 3.
`m_post`	An integer. Optionally paramter only used in non-conjugate case. Default is 3.
`iters`	An integer. Number of iterations. Default is 5000.
`burnin`	An integer. Amount of burn-in. Default is 200.
`shuffled`	A logical. Whether or not to shuffle the data. Default is true.

Details

Performs `iters` iterations of Algorithm 2 (in conjugate case) or Algorithm 8 (in non-conjugate case) from Neal(2000) to generate possible clusters for the data in `Xdata`, using the model in `model`, with concentration parameter `alpha`. In the 1D case, `Xdata` is assumed to be a 1D array of floats. In the 2D case, `Xdata` is assumed to be a dxN array of floats, where the data is d-dimensional and N is the number of datapoints. Returns a list of states. The elements of the list are all states post-burnin iteration, with the default being a `burnin` of 200. By default, this array is shuffled so that it may be used to approximate I.I.D draws from the posterior.

A single state from the returned list of states has fields `data` and `clusters`. `data` is a dataframe consisting of the `Xdata` and their cluster labels. `clusters` is a data.table (is the user has the data.table package loaded) or a list.

If clusters is a data.table, each row refers to a cluster. Columns are the cluster label, the population, and the rest of the columns are parameters.

If clusters is a list, each element of the list refers to a clsuter, clusters[[i]] is a list containing of the above information for cluster i as elements. Each single item in clusters is a list with fields `cluster`, `population`, and `params`. E.g. clusters[[1]]$population is the population of cluster 1. The params field (clusterInfo[[i]]$params) is itself a list of each of the parameters

To see a formatted summary of all the clusters in a given state use the `dmm.summarize(clusters)` function.

To see a plot of the labled data in a given state use the `dmm.plot(data)` function.

Value

A list of states (i.e. `state = states[[i]]`). A state is itself a list. A state has two fields: `labeledData` and `clusterInfo`.

`labeledData` is a data.frame of the `Xdata` data points and their cluster labels. `clusterInfo` is either a list or a data.table (if the data.table package is loaded by the user). It conatins (1) cluster labels, (2) the number of data points (i.e. population) of each cluster, and (3) all of the parameters for each cluster.

Examples

```
dmm.setup()
model <- dmm.BaseModel(data = Xdata)
states <- dmm.cluster(model, Xdata)
```

---

dmm.cluster.BaseModel *Use a Dirichlet Mixture Model on data to get cluster labels and cluster parameter values.*

Description

Use a Dirichlet Mixture Model on data to get cluster labels and cluster parameter values.

Usage

```
dmm.cluster.BaseModel(model, Xdata, alpha=1.0, iters=5000, burnin=200, shuffled=TRUE)
```

Arguments

`model`	An object returned by `dmm.model()`.
`Xdata`	A 1D array of length N (univariate case) or 2D array of size N-by-d (mulit-variate case), where d is the dimensionailty of the data and N is the number of observations.
`alpha`	A float. The concentration parameter. Default is 1.0.
`iters`	An integer. Number of iterations. Default is 5000.
`burnin`	An integer. Amount of burn-in. Default is 200.
`shuffled`	A logical. Whether or not to shuffled the data. Default is true.
`m_prior`	An integer. Optionally paramter only used in non-conjugate case. Default is 3.
`m_post`	An integer. Optionally paramter only used in non-conjugate case. Default is 3.

Details

Performs `iters` iterations of Algorithm 2 (in conjugate case) or Algorithm 8 (in non-conjugate case) from Neal(2000) to generate possible clusters for the data in `Xdata`, using the model in `model`, with concentration parameter `alpha`. In the 1D case, `Xdata` is assumed to be a 1D array of floats. In the 2D case, `Xdata` is assumed to be a dxN array of floats, where the data is d-dimensional and N is the number of datapoints. Returns a list of states. The elements of the list are all states post-burnin iteration, with the default being a `burnin` of 200. By default, this array is shuffled so that it may be used to approximate I.I.D draws from the posterior.

A single state from the returned list of states has fields `data` and `clusters`. `data` is a dataframe consisting of the `Xdata` and their cluster labels. `clusters` is a data.table (is the user has the data.table package loaded) or a list.

If clusters is a data.table, each row refers to a cluster. Columns are the cluster label, the population, and the rest of the columns are parameters.

If clusters is a list, each element of the list refers to a clsuter, clusters[[i]] is a list containing of the above information for cluster i as elements. Each single item in clusters is a list with fields `cluster`, `population`, and `params`. E.g. clusters[[1]]$population is the population of cluster 1. The params field (clusters[[i]]$params) is itself a list of each of the parameters

To see a formatted summary of all the clusters in a given state use the `dmm.summarize(clusters)` function.

To see a plot of the labled data in a given state use the `dmm.plot(data)` function.

Value

A list of states (i.e. `state = states[[i]]`). A state is itself a list. A state has two fields: `data` and `clusters`.

`data` is a data.frame of the `Xdata` data points and their cluster labels. `clusters` is either a list or a data.table (if the data.table package is loaded by the user). It conatins (1) cluster labels, (2) the number of data points (i.e. population) of each cluster, and (3) all of the parameters for each cluster.

Examples

```
dmm.setup()
model <- dmm.BaseModel(data = Xdata)
states <- dmm.cluster(model, Xdata)
```

---

dmm.cluster.JConjugateModel

Use a Dirichlet Mixture Model on data to get cluster labels and cluster parameter values.

Description

Use a Dirichlet Mixture Model on data to get cluster labels and cluster parameter values.

Usage

```
## S3 method for class 'JConjugateModel'
dmm.cluster(model, Xdata, alpha = 1, m_prior = 3,
  m_post = 3, iters = 5000, burnin = 200, shuffled = TRUE)
```

Arguments

`model`	An object returned by `dmm.model()`.
`Xdata`	A 1D array of length N (univariate case) or 2D array of size N-by-d (mulitvariate case), where d is the dimensionailty of the data and N is the number of observations.
`alpha`	A float. The concentration parameter. Default is 1.0.

m_prior	An integer. Optionally paramter only used in non-conjugate case. Default is 3.
m_post	An integer. Optionally paramter only used in non-conjugate case. Default is 3.
iters	An integer. Number of iterations. Default is 5000.
burnin	An integer. Amount of burn-in. Default is 200.
shuffled	A logical. Whether or not to shuffled the data. Default is true.

Details

Performs `iters` iterations of Algorithm 2 (in conjugate case) or Algorithm 8 (in non-conjugate case) from Neal(2000) to generate possible clusters for the data in `Xdata`, using the model in `model`, with concentration parameter `alpha`. In the 1D case, `Xdata` is assumed to be a 1D array of floats. In the 2D case, `Xdata` is assumed to be a dxN array of floats, where the data is d-dimensional and N is the number of datapoints. Returns a list of states. The elements of the list are all states post-burnin iteration, with the default being a `burnin` of 200. By default, this array is shuffled so that it may be used to approximate I.I.D draws from the posterior.

A single state from the returned list of states has fields `data` and `clusters`. `data` is a dataframe consisting of the `Xdata` and their cluster labels. `clusters` is a data.table (is the user has the data.table package loaded) or a list.

If clusters is a data.table, each row refers to a cluster. Columns are the cluster label, the population, and the rest of the columns are parameters.

If clusters is a list, each element of the list refers to a clsuter, clusters[[i]] is a list containing of the above information for cluster i as elements. Each single item in clusters is a list with fields `cluster`, `population`, and `params`. E.g. clusters[[1]]$population is the population of cluster 1. The params field (clusters[[i]]$params) is itself a list of each of the parameters

To see a formatted summary of all the clusters in a given state use the `dmm.summarize(clusters)` function.

To see a plot of the labled data in a given state use the `dmm.plot(data)` function.

Value

A list of states (i.e. `state = states[[i]]`). A state is itself a list. A state has two fields: `data` and `clusters`.

`data` is a data.frame of the `Xdata` data points and their cluster labels. `clusters` is either a list or a data.table (if the data.table package is loaded by the user). It conatins (1) cluster labels, (2) the number of data points (i.e. population) of each cluster, and (3) all of the parameters for each cluster.

Examples

```
dmm.setup()
dmm.addfile(filename)
model <- dmm.JConjugateModel(pdf_name, sample_name, marg_name, params)
states <- dmm.cluster(model, Xdata)
```

`dmm.cluster.JNonConjugateModel` *Use a Dirichlet Mixture Model on data to get cluster labels and cluster parameter values.*

Description

Use a Dirichlet Mixture Model on data to get cluster labels and cluster parameter values.

Usage

```
## S3 method for class 'JNonConjugateModel'
dmm.cluster(model, Xdata, alpha = 1,
  m_prior = 3, m_post = 4, iters = 5000, burnin = 200,
  shuffled = TRUE)
```

Arguments

`model`	An object returned by `dmm.model()`.
`Xdata`	A 1D array of length N (univariate case) or 2D array of size N-by-d (mulit-variate case), where d is the dimensionailty of the data and N is the number of observations.
`alpha`	A float. The concentration parameter. Default is 1.0.
`m_prior`	An integer. Optionally paramter only used in non-conjugate case. Default is 3.
`m_post`	An integer. Optionally paramter only used in non-conjugate case. Default is 3.
`iters`	An integer. Number of iterations. Default is 5000.
`burnin`	An integer. Amount of burn-in. Default is 200.
`shuffled`	A logical. Whether or not to shuffled the data. Default is true.

Details

Performs `iters` iterations of Algorithm 2 (in conjugate case) or Algorithm 8 (in non-conjugate case) from Neal(2000) to generate possible clusters for the data in `Xdata`, using the model in `model`, with concentration parameter `alpha`. In the 1D case, `Xdata` is assumed to be a 1D array of floats. In the 2D case, `Xdata` is assumed to be a dxN array of floats, where the data is d-dimensional and N is the number of datapoints. Returns a list of states. The elements of the list are all states post-burnin iteration, with the default being a `burnin` of 200. By default, this array is shuffled so that it may be used to approximate I.I.D draws from the posterior.

A single state from the returned list of states has fields `data` and `clusters`. `data` is a dataframe consisting of the `Xdata` and their cluster labels. `clusters` is a data.table (is the user has the data.table package loaded) or a list.

If clusters is a data.table, each row refers to a cluster. Columns are the cluster label, the population, and the rest of the columns are parameters.

If clusters is a list, each element of the list refers to a clsuter, clusters[[i]] is a list containing of the above information for cluster i as elements. Each single item in clusters is a list with fields `cluster`, `population`, and `params`. E.g. clusters[[1]]$population is the population of cluster 1. The params field (clusters[[i]]$params) is itself a list of each of the parameters

To see a formatted summary of all the clusters in a given state use the `dmm.summarize(clusters)` function.

To see a plot of the labled data in a given state use the `dmm.plot(data)` function.

Value

A list of states (i.e. `state = states[[i]]`). A state is itself a list. A state has two fields: `data` and `clusters`.

`data` is a data.frame of the `Xdata` data points and their cluster labels. `clusters` is either a list or a data.table (if the data.table package is loaded by the user). It conatins (1) cluster labels, (2) the number of data points (i.e. population) of each cluster, and (3) all of the parameters for each cluster.

Examples

```
dmm.setup()
dmm.addfile(filename)
model <- dmm.JNonConjugateModel(pdf_name, sample_name, params)
states <- dmm.cluster(model, Xdata)
```

---

dmm.cluster.RModel *Use a Dirichlet Mixture Model on data to get cluster labels and cluster parameter values.*

Description

Use a Dirichlet Mixture Model on data to get cluster labels and cluster parameter values.

Usage

```
## S3 method for class 'RModel'
dmm.cluster(model, Xdata, alpha = 1, m_prior = 3,
  m_post = 3, iters = 5000, burnin = 200, shuffled = TRUE)
```

Arguments

`model`	An object returned by `dmm.model()`.
`Xdata`	A 1D array of length N (univariate case) or 2D array of size N-by-d (mulitvariate case), where d is the dimensionailty of the data and N is the number of observations. Use a Dirichlet Mixture Model on data to get cluster labels and cluster parameter values.
`alpha`	A float. The concentration parameter. Default is 1.0.
`m_prior`	An integer. Optionally paramter only used in non-conjugate case. Default is 3.
`m_post`	An integer. Optionally paramter only used in non-conjugate case. Default is 3.
`iters`	An integer. Number of iterations. Default is 5000.
`burnin`	An integer. Amount of burn-in. Default is 200.
`shuffled`	A logical. Whether or not to shuffled the data. Default is true.
`model`	An object returned by `dmm.model()`.
`Xdata`	A 1D array of length N (univariate case) or 2D array of size N-by-d (mulitvariate case), where d is the dimensionailty of the data and N is the number of observations.

Details

Performs `iters` iterations of Algorithm 2 (in conjugate case) or Algorithm 8 (in non-conjugate case) from Neal(2000) to generate possible clusters for the data in `Xdata`, using the model in `model`, with concentration parameter `alpha`. In the 1D case, `Xdata` is assumed to be a 1D array of floats. In the 2D case, `Xdata` is assumed to be a dxN array of floats, where the data is d-dimensional and N is the number of datapoints. Returns a list of states. The elements of the list are all states post-burnin iteration, with the default being a `burnin` of 200. By default, this array is shuffled so that it may be used to approximate I.I.D draws from the posterior.

A single state from the returned list of states has fields `data` and `clusters`. `data` is a dataframe consisting of the `Xdata` and their cluster labels. `clusters` is a data.table (is the user has the data.table package loaded) or a list.

If clusters is a data.table, each row refers to a cluster. Columns are the cluster label, the population, and the rest of the columns are parameters.

If clusters is a list, each element of the list refers to a clsuter, clusters[[i]] is a list containing of the above information for cluster i as elements. Each single item in clusters is a list with fields `cluster`, `population`, and `params`. E.g. clusters[[1]]$population is the population of cluster 1. The params field (clusters[[i]]$params) is itself a list of each of the parameters

To see a formatted summary of all the clusters in a given state use the `dmm.summarize(clusters)` function.

To see a plot of the labled data in a given state use the `dmm.plot(data)` function.

Value

A list of states (i.e. `state = states[[i]]`). A state is itself a list. A state has two fields: `data` and `clusters`.

`data` is a data.frame of the `Xdata` data points and their cluster labels. `clusters` is either a list or a data.table (if the data.table package is loaded by the user). It conatins (1) cluster labels, (2) the number of data points (i.e. population) of each cluster, and (3) all of the parameters for each cluster.

---

dmm.JConjugateModel *Create a conjugate model using Julia fucntions*

Description

Create an model object to be used in the `dmm.cluster` function, using user given Julia functions. Must call `dmm.addfile` to import files, in which the Julia functions are stored, before using `dmm.cluster` on a JModel. Functions `dmm.JModel` and `dmm.model` are alternatives.

Usage

```
dmm.addfile(filename)
model <- dmm.JConjugateModel(pdf_name, sample_name, marg_name, params)
```

Arguments

`pdf_name`	A string. The name of the Julia function in `filename` that returns the probability density function likelihood. The function should be of the form `pdf_name(y::Float64, theta::Tup` or `pdf_name(y::Array{Float64,1}, theta::Tuple, params::Tuple)`.

sample_name	A string. The name of the Julia function in filename that returns the sample posterior function. The function should be of the form sample_name(y::Float64, params::Tuple) sample_name(y::Array{Float64,1}, params::Tuple) or sample_name(y::Array{Float64,2}
marg_name	A string. The name of the Julia function in filename that returns the marginal likelihood. The function should be of the form marg_name(y::Float64, params::Tuple).
params	A list of all hyperparameters needed for the above three functions.
isconjugate	A logical. TRUE (default) if the user specfied model is conjugate, FALSE if not.

Value

A model object of type JModel which can be passed to dmm.cluster.

Examples

```
dmm.addfile(filename)
# The following all make conjugate models using Julia functions
model <- dmm.model(pdf_name, sample_name, marg_name, params, isconjugate=TRUE)
model <- dmm.Jmodel(pdf_name, sample_name, marg_name, params, isconjugate=TRUE)
model <- dmm.JConjugateModel(pdf_name, sample_name, marg_name, params)
```

---

dmm.JModel *Create a model using Julia fucntions*

Description

Create an model object to be used in the dmm.cluster function, using user given Julia functions. Must call dmm.addfile to import files, in which the Julia functions are stored, before using dmm.cluster on a JModel. Functions dmm.JConjugateModel and dmm.JNonConjugateModel are alternatives.

Usage

```
\code{dmm.addfile(filename)}
\code{model <- dmm.model(pdf_name, sample_name, marg_name, params, isconjugate=TRUE)}
\code{model <- dmm.Jmodel(pdf_name, sample_name, marg_name, params, isconjugate=TRUE)}
\code{model <- dmm.JConjugateModel(pdf_name, sample_name, marg_name, params)}
\code{model <- dmm.JNonConjugateModel(pdf_name, sample_name, params)}
```

Arguments

pdf_name	A string. The name of the Julia function in filename that returns the probability density function likelihood. The function should be of the form pdf_name(y::Float64, theta::Tup or pdf_name(y::Array{Float64,1}, theta::Tuple, params::Tuple).
sample_name	A string. The name of the Julia function in filename that returns the sample posterior function for conjugate case or the sample prior for nonconjugate case. The function should be of the form sample_name(y::Float64, params::Tuple), sample_name(y::Array{Float64,1}, params::Tuple) or sample_name(y::Array{Float64,2}
marg_name	A string. For conjugate case only. The name of the Julia function in filename that returns the marginal likelihood. The function should be of the form marg_name(y::Float64, pa
params	A list of all hyperparameters needed for the above three functions.
isconjugate	A logical. TRUE (default) if the user specfied model is conjugate, FALSE if not.

Details

marg_name is only requried for conjugate models.

Value

A model object of type JModel which can be passed to dmm.cluster.

Examples

```
dmm.addfile(filename)
# The following all make models using Julia functions
model <- dmm.model(pdf_name, sample_name, marg_name, params, isconjugate=TRUE)
model <- dmm.Jmodel(pdf_name, sample_name, marg_name, params, isconjugate=TRUE)
model <- dmm.JConjugateModel(pdf_name, sample_name, marg_name, params)
model <- dmm.JNonConjugateModel(pdf_name, sample_name, params)
```

---

dmm.JNonConjugateModel

Create a nonconjugate model using Julia fucntions

Description

Create an model object to be used in the dmm.cluster function, using user given Julia functions. Must call dmm.addfile to import files, in which the Julia functions are stored, before using dmm.cluster on a JModel. Functions dmm.JModel and dmm.model are alternatives..

Usage

```
dmm.addfile(filename)
model <- dmm.model(pdf_name, sample_name, marg_name, params, isconjugate=TRUE)
```

Arguments

pdf_name	A string. The name of the Julia function in filename that returns the probability density function likelihood. The function should be of the form pdf_name(y::Float64, theta::Tup or pdf_name(y::Array{Float64,1}, theta::Tuple, params::Tuple).
sample_name	A string. The name of the Julia function in filename that returns the sample prior. The function should be of the form sample_name(y::Float64, params::Tuple), sample_name(y::Array{Float64,1}, params::Tuple) or sample_name(y::Array{Float64,2}
params	A list of all hyperparameters needed for the above three functions.

Value

A model object of type JModel which can be passed to dmm.cluster.

Examples

```
dmm.addfile(filename)
# The following all make nonconjugate models using Julia functions
model <- dmm.model(pdf_name, sample_name, params, isconjugate=FALSE)
model <- dmm.Jmodel(pdf_name, sample_name, params, isconjugate=FALSE)
model <- dmm.JNonConjugateModel(pdf_name, sample_name, params)
```

---

dmm.model *Create a model*

Description

Make a model object to be passed to dmm.cluster. If not passed any argurments a conjugate multivariate normal likelihood model with default parameters will be used.

Usage

```
dmm.model(...)
```

Details

Depending on what argurments are passed to dmm.model() either a BaseModel (see dmm.BaseModel), a RModel (see dmm.RModel), or JModel (see dmm.JModel) will be made.

NOTE: The use of RModels is not avaible in the current version of jDirichletMixtureModels

Value

A model object which can be passed to dmm.cluster.

Examples

```
## Not run:
# Example of using a multivariate normal conjugate prior (the default)
model <- dmm.model()

# Example of using a bulit-in conjugate prior
model <- dmm.model(typename, params)

# Using a user specified R functions
model <- dmm.model(pdf_fct, sample_fct, marg_fct, params, isconjugate=TRUE)

# Using a user specified Julia functions located in a .jl file called filename
dmm.addfile(filename)
model <- dmm.model(pdf_name, sample_name, marg_name, params, isconjugate=TRUE)

## End(Not run)
```

dmm.model.character *Create a model object*

Description

If a string is passed as the first input to dmm.model it will create and return either a BaseModel (if passed one string, plus an optional params) (see dmm.BaseModel), or a JModel (see dmm.JModel)

Usage

```
## S3 method for class 'character'
dmm.model(...)
```

dmm.model.function *Create a model object*

Description

If a function is passed as the first input to dmm.model it will create and return an Rmodel (see dmm.RModel).

Usage

```
## S3 method for class 'function'
dmm.model(...)
```

dmm.model.NULL *Create a model object*

Description

If nothing is passed to dmm.model it will create and return a BaseModel using a multivariate normal conjugate model with default parameters (see dmm.BaseModel).

Usage

```
## S3 method for class 'NULL'
dmm.model(...)
```

dmm.plot *Plot labeledData for a state returned by dmm*

Description

Plot labeledData for a state returned by dmm

Usage

```
dmm.plot(labeledData)

Given the data from a single state returned by dmm.cluster(...), plot it. Can do 2D, 1D, or 3D plots.
\code{ggplot2} recommanded for 2D plots. \code{scatterplot3d} requried for 3D plots.
```

Arguments

labeledData	The data from a single state. Given states <- dmm.cluster(...), this input is states[[i]]$data.

Examples

```
states <- dmm.cluster(model,Xdata,...)
 dmm.plot(states[[1]]$data)
```

dmm.RModel *Create a model using R fucntions*

Description

NOTE: This function is not avaible in the current version of jDirichletMixtureModels. Create an model object to be used in the dmm.cluster function, using user given R functions.

Usage

```
dmm.RModel(pdf_func, sample_func = NULL, marg_func = NULL, params,
  isconjugate = TRUE)
```

Arguments

pdf_func	A function that takes as input (data, likelihoodparams, params). It returns the value of the probability density function likelihood at (data,likelihoodparams) given params.
params	A list of all hyperparameters needed for the above three functions.
isconjugate	A logical. TRUE (default) if the user specfied model is conjugate, FALSE if not.
sample_fct	Optional for nonconjugate models. A function takes as input (data, params). It returns the value of the sample posterior function at data given params.
marg_fct	Only needed for conjugate models. A function takes as input (data, params). It returns the value of the marginal likelihood funciton at data given params..

Details

@usage model <- dmm.Rmodel(pdf_fct, sample_fct, marg_fct, params, isconjugate=TRUE)

Value

A model object of type RModel which can be passed to `dmm.cluster`.

#@export

`dmm.setup` *Do initial setup for jDirichletMixtureModels package.*

Description

`dmm.setup` does the initial setup for jDirichletMixtureModels package.

Usage

```
dmm.setup()
```

Arguments

`...`	arguments passed to `JuliaCall::julia_setup`. Most noteable:the first input should be the path to the folder in which the Julia bin file is located on the user's machine. It will default to using the macOSX path for the most recent Julia version.

Examples

```
## Not run:
dmm.setup()

## End(Not run)
```

`dmm.state` *Formats a single DMM OutputState Julia object into a list of labeled data and list of cluster info*

Description

Formats a single DMM OutputState Julia object into a list of labeled data and list of cluster info

Usage

```
dmm.state(juliastate, paramnames)
```

`dmm.stateAsTable` *Formats a single DMM OutputState Julia object into a list of labeled data and data.table of cluster info*

Description

Formats a single DMM OutputState Julia object into a list of labeled data and data.table of cluster info

Usage

```
dmm.stateAsTable(juliastate, paramnames)
```

`dmm.states` *Constructing list of all states from dmm.cluster run (excluding burnin).*

Description

Constructing list of all states from dmm.cluster run (excluding burnin).

Usage

```
dmm.states(juliastates, paramnames = NULL)
```

Arguments

`juliastates`	The object returned by Julia code
`paramnames`	Optionally. A list of the parameter names. Returned by Julia code for most bulit-in models.

Details

Each item in the list (i.e. `state = states[[i]]`) is a state. A state is also a list. A state has two fields: `data` and `clusters`.

`data` is a data.frame of the data points and their cluster labels. `clusters` is either a list or a data.table (if the data.table package is loaded by the user). It conatins (1) cluster labels, (2) the number of data points (i.e. population) of each cluster, and (3) all of the parameters for each cluster.

If clusters is a data.table, each row refers to a cluster. Columns are the cluster label, the population, and the rest of the columns are parameters.

If clusters is a list, each element of the list refers to a clsuter, clusters[[i]] is a list containing of the above information for cluster i as elements. E.g. clusters[[1]]$population is the population of cluster 1. The params field (clusters[[i]]$params) is itself a list of each of the parameters

dmm.summarize *Summerize given a single state's cluster info*

Description

Summerize given a single state's cluster info

Usage

```
dmm.summarize(clusterInfo)
```

Arguments

clusterInfo	A list or data.table. Given states <- dmm.cluster(...), this input is states[[i]]$clusters

Examples

```
states <- dmm.cluster(model, data)
dmm.summarize(states[[1]]$clusters)
```

dmm.summarize.data.table *Summerize given a single state's cluster info*

Description

Summerize given a single state's cluster info

Usage

```
dmm.summarize(clusterInfo)
```

Arguments

clusterInfo	A data.table. Given states <- dmm.cluster(...), this input is states[[i]]$clusters

Examples

```
states <- dmm.cluster(model, data)
dmm.summarize(states[[1]]$clusters)
```

dmm.summarize.list *Summerize given a single state's cluster info*

Description

Summerize given a single state's cluster info

Usage

```
dmm.summarize(clusterInfo)
```

Arguments

clusterInfo	A list. Given states <- dmm.cluster(...), this input is states[[i]]$clusters

Examples

```
states <- dmm.cluster(model, data)
dmm.summarize(states[[1]]$clusters)
```

mouse *Mouse Data*

Description

Clusters that look like Mickey mouse's head.

Usage

```
data(mouse, package = "jDirichletMixtureModels")
```

Format

An object of class `matrix` with 500 rows and 2 columns.

Source

mouse.csv

Examples

```
## Not run:
Xdata <- data(mouse, package = "jDirichletMixtureModels")
states <- dmm.cluster(model, Xdata)
dmm.plot(states[[1]]$labeledData)

## End(Not run)
```

syntheticGaussian1 *Synthetic Gaussian Cluster Data 1*

Description

Synthetic 2-d data with N=5000 vectors and k=15 Gaussian clusters with different degree of cluster overlapping

Usage

```
data(syntheticGaussian1, package = "jDirichletMixtureModels")
```

Format

An object of class `matrix` with 5000 rows and 2 columns.

Source

Clustering basic benchmark

References

P. Fränti and O. Virmajoki, "Iterative shrinking method for clustering problems", Pattern Recognition, 39 (5), 761-765, May 2006.

Examples

```
## Not run:
Xdata <- data(syntheticGaussian1, package = "jDirichletMixtureModels")
states <- dmm.cluster(model, Xdata)
dmm.plot(states[[1]]$labeledData)

## End(Not run)
```

syntheticGaussian2 *Synthetic Gaussian Cluster Data 2*

Description

Synthetic 2-d data with N=5000 vectors and k=15 Gaussian clusters with different degree of cluster overlapping

Usage

```
data(syntheticGaussian2, package = "jDirichletMixtureModels")
```

Format

An object of class `matrix` with 5000 rows and 2 columns.

Source

Clustering basic benchmark

References

P. Fränti and O. Virmajoki, "Iterative shrinking method for clustering problems", Pattern Recognition, 39 (5), 761-765, May 2006.

Examples

```
## Not run:
Xdata <- data(syntheticGaussian2, package = "jDirichletMixtureModels")
states <- dmm.cluster(model, Xdata)
dmm.plot(states[[1]]$labeledData)

## End(Not run)
```

Index