

# Package ‘KSD’

October 12, 2022

**Type** Package

**Title** Goodness-of-Fit Tests using Kernelized Stein Discrepancy

**Version** 1.0.1

**Date** 2021-01-11

**Description** An adaptation of Kernelized Stein Discrepancy, this package provides a goodness-of-fit test of whether a given i.i.d. sample is drawn from a given distribution. It works for any distribution once its score function (the derivative of log-density) can be provided. This method is based on “A Kernelized Stein Discrepancy for Goodness-of-fit Tests and Model Evaluation” by Liu, Lee, and Jordan, available at [arXiv:1602.03253](https://arxiv.org/abs/1602.03253).

**License** MIT + file LICENSE

**LazyData** TRUE

**RoxygenNote** 7.1.1

**Imports** pryr, graphics, stats

**Suggests** datasets, ggplot2, gridExtra, mclust, mvtnorm

**NeedsCompilation** no

**Author** Min Hyung Kang [aut, cre],  
Qiang Liu [aut]

**Maintainer** Min Hyung Kang <Minhyung.Daniel.Kang@gmail.com>

**Repository** CRAN

**Date/Publication** 2021-01-11 08:50:16 UTC

## R topics documented:

demo_gmm . . . . .	2
demo_gmm_multi . . . . .	2
demo_iris . . . . .	2
demo_normal_performance . . . . .	3
demo_simple_gamma . . . . .	3
demo_simple_gaussian . . . . .	4
gmm . . . . .	4
KSD . . . . .	5

likelihoodgmm . . . . .	6
perturbgmm . . . . .	7
plotgmm . . . . .	8
posteriorgmm . . . . .	8
rgmm . . . . .	9
scorefunctiongmm . . . . .	10

<b>Index</b>	<b>11</b>
--------------	-----------

---

demo_gmm	<i>Tests 1-dimensional Gaussian Mixture Models.</i>
----------	---

---

### Description

Tests 1-dimensional Gaussian Mixture Models.

### Usage

demo\_gmm()

---

demo_gmm_multi	<i>Tests multidimensional Gaussian Mixture Models.</i>
----------------	--

---

### Description

Tests multidimensional Gaussian Mixture Models.

### Usage

demo\_gmm\_multi()

---

demo_iris	<i>Fits Gaussian Mixture model and computes the KSD value for the model</i>
-----------	---

---

### Description

We fit a Gaussian Mixture Model for a given dataset (Fisher's Iris), and we compute the KSD P-value on the hold-out test dataset. User may tune the parameters and observe the change in results. Reports average of p-values obtained during each k-fold. It also plots the contour for each k-fold iteration if only 2 dimensions of data are used. If a vector is specified for nClust, the code tries each element as the number of clusters and reports the optimal parameter by choosing one with highest p-value.

**Usage**

```
demo_iris(cols = c(1, 2), nClust = 3, kfold = 5)
```

**Arguments**

cols : Columns of iris data set to use. If 2 dimensions, plots the contour for each k-fold.

nClust : Number of clusters want to estimate with If vector, use each element as number of clusters and reports the optimal number.

kfold : Number of k to use for k-fold

---

demo\_normal\_performance

*Shows KSD p value change with respect variation in noise*

---

**Description**

We generate a standard normal distribution, and add varying gaussian noise to this dataset and see the change in pvalues.

**Usage**

```
demo_normal_performance()
```

---

demo\_simple\_gamma

*Tests 1-dimensional Gamma Distribution with customized parameters*

---

**Description**

We generate a gamma distribution with given parameters, and add gaussian noise to this dataset. We then compute the score of each dataset for the original true distribution.

**Usage**

```
demo_simple_gamma(  
  trueshape = 10,  
  truescale = 3,  
  noisemu = 5,  
  noisesd = 2,  
  n = 100  
)
```

**Arguments**

trueshape	shape of true gamma distribution
truescale	scale of true gamma distribution
noisemu	mean of gaussian noise to add
noisesd	standard deviation of gaussian noise to add
n	number of samples to generate

---

demo\_simple\_gaussian *Tests 1-dimensional Gaussian Distribution with customized parameters*

---

**Description**

We generate a gaussian distribution with given parameters, and add noise to this dataset. We then compute the score of each dataset for the original true distribution.

**Usage**

```
demo_simple_gaussian(truemu = 5, truesd = 1, noisemu = 0, noisesd = 2, n = 100)
```

**Arguments**

truemu	mean of true distribution
truesd	standard deviation of true distribution
noisemu	mean of gaussian noise to add
noisesd	standard deviation of gaussian noise to add
n	number of samples to generate

---

gmm *Returns a Gaussian Mixture Model*

---

**Description**

Returns a Gaussian Mixture Model

**Usage**

```
gmm(nComp = NULL, mu = NULL, sigma = NULL, weights = NULL, d = NULL)
```

**Arguments**

nComp	(scalar) : number of components
mu	(d by k): mean of each component
sigma	(d by d by k): covariance of each component
weights	(1 by k) : mixing weight of each proportion (optional)
d	: number of dimensions of vector (optional)

**Value**

model : A Gaussian Mixture Model generated from the given parameters

**Examples**

```
# Default 1-d gaussian mixture model
model <- gmm()

# 1-d Gaussian mixture model with 3 components
model <- gmm(nComp = 3)

# 3-d Gaussian mixture model with 3 components, with specified mu,sigma and weights
mu <- matrix(c(1,2,3,2,3,4,5,6,7),ncol=3)
sigma <- array(diag(3),c(3,3,3))
model <- gmm(nComp = 3, mu = mu, sigma=sigma, weights = c(0.2,0.4,0.4), d = 3)
```

KSD

*Estimate Kernelized Stein Discrepancy (KSD)***Description**

Estimate kernelized Stein discrepancy (KSD) using U-statistics, and use bootstrap to test  $H_0: x_i$  is drawn from  $p(X)$  (via  $KSD=0$ ).

**Usage**

```
KSD(x, score_function, kernel = "rbf", width = -1, nboot = 1000)
```

**Arguments**

x	Sample of size Num_Instance x Num_Dimension
score_function	$(\nabla_x \log p(x))$ Score funtion : takes x as input and output a column vector of size Num_Instance X Dimension. User may use pryr package to pass in a function that only takes in dataset as parameter, or user may also pass in computed score for a given dataset.
kernel	Type of kernel (default = 'rbf')
width	Bandwidth of the kernel (when width = -1 or 'median', set it to be the median distance between data points)
nboot	Bootstrap sample size

**Value**

A list which includes the following variables :

- "ksd" : Estimated Kernelized Stein Discrepancy (KSD)
- "p" : p-Value for rejecting the null hypothesis that  $\text{ksd} = 0$
- "bootstrapSamples" : the bootstrap sample
- "info": other information, including : bandwidth, M, nboot, ksd\_V

**Examples**

```
# Pass in a dataset generated by Gaussian distribution,
# use pryr package to pass in score function
model <- gmm()
X <- rgmm(model, n=100)
score_function = pryr::partial(scorefunctiongmm, model=model)
result <- KSD(X,score_function=score_function)

# Pass in a dataset generated by Gaussian distribution,
# pass in computed score rather than score function
model <- gmm()
X <- rgmm(model, n=100)
score_function = scorefunctiongmm(model=model, X=X)
result <- KSD(X,score_function=score_function)

# Pass in a dataset generated by Gaussian distribution,
# pass in computed score rather than score function
# Use median_heuristic by specifying width to be -2.0
model <- gmm()
X <- rgmm(model, n=100)
score_function = pryr::partial(scorefunctiongmm, model=model)
result <- KSD(X,score_function=score_function, 'rbf',-2.0)

# Pass in a dataset generated by specific Gaussian distribution,
# pass in computed score rather than score function
# Use median_heuristic by specifying width to be -2.0
model <- gmm()
X <- rgmm(model, n=100)
score_function = pryr::partial(scorefunctiongmm, model=model)
result <- KSD(X,score_function=score_function, 'rbf',-2.0)
```

---

likelihoodgmm

*Calculates the likelihood for a given dataset for a GMM*


---

**Description**

Calculates the likelihood for a given dataset for a GMM

**Usage**

```
likelihoodgmm(model = NULL, X = NULL)
```

**Arguments**

`model` : The Gaussian Mixture Model

`X` (n by d): The dataset of interest, where n is the number of samples and d is the dimension

**Value**

P (n by k) : The likelihood of each dataset belonging to each of the k component

**Examples**

```
# compute likelihood for a default 1-d gaussian mixture model
# and dataset generated from it
model <- gmm()
X <- rgmm(model)
p <- likelihoodgmm(model=model, X=X)
```

---

`perturbgmm`

*Returns a perturbed model of given GMM*

---

**Description**

Returns a perturbed model of given GMM

**Usage**

```
perturbgmm(model = NULL)
```

**Arguments**

`model` : The base Gaussian Mixture Model

**Value**

`perturbedModel` : Perturbed model with added noise to the supplied GMM

**Examples**

```
#Add noise to default 1-d gaussian mixture model
model <- gmm()
noisymodel <- perturbgmm(model)
```

plotgmm

*Plots histogram for 1-d GMM given the dataset*

---

**Description**

Plots histogram for 1-d GMM given the dataset

**Usage**

```
plotgmm(data, mu = NULL)
```

**Arguments**

`data` (n by 1): The dataset of interest, where n is the number of samples.  
`mu` : True mean of the GMM (optional)

**Examples**

```
# Plot pdf histogram for a given dataset
model <- gmm()
X <- rgmm(model)
plotgmm(data=X)

# Plot pdf histogram for a given dataset, with lines that indicate the mean
model <- gmm()
mu <- model$mu
X <- rgmm(model)
plotgmm(data=X, mu=mu)
```

---

posteriorgmm*Calculates the posterior probability for a given dataset for a GMM*

---

**Description**

Calculates the posterior probability for a given dataset for a GMM

**Usage**

```
posteriorgmm(model = NULL, X = NULL)
```

**Arguments**

`model` : The Gaussian Mixture Model  
`X` (n by d): The dataset of interest, where n is the number of samples and d is the dimension



**Value**

P (n by k) : The posterior probability of each dataset belonging to each of the k component

**Examples**

```
# compute posterior probability for a default 1-d gaussian mixture model
# and dataset generated from it
model <- gmm()
X <- rgmm(model)
p <- posteriorgmm(model=model, X=X)
```

---

rgmm

*Generates dataset from Gaussian Mixture Model*

---

**Description**

Generates dataset from Gaussian Mixture Model

**Usage**

```
rgmm(model = NULL, n = 100)
```

**Arguments**

model : Gaussian Mixture Model defined by gmm()  
n : number of samples desired

**Value**

data (n by d): Random dataset generated from given the Gaussian Mixture Model

**Note**

Requires library mvtnorm

**Examples**

```
#Generate 100 samples from default gaussian mixture model
model <- gmm()
X <- rgmm(model)

#Generate 300 samples from 3-d gaussian mixture model
model <- gmm(d=3)
X <- rgmm(model,n=300)
```

---

scorefunctiongmm      *Score function for given GMM : calculates score function  $d\log p(x)/dx$  for a given Gaussian Mixture Model*

---

**Description**

Score function for given GMM : calculates score function  $d\log p(x)/dx$  for a given Gaussian Mixture Model

**Usage**

```
scorefunctiongmm(model = NULL, X = NULL)
```

**Arguments**

model                   : The Gaussian Mixture Model  
X                        (n by d): The dataset of interest, where n is the number of samples and d is the dimension

**Value**

y : The score computed by the given function

**Examples**

```
# Compute score for a given gaussianmixture model and dataset  
model <- gmm()  
X <- rgmm(model)  
score <- scorefunctiongmm(model=model, X=X)
```

# Index

demo\_gmm, [2](#)  
demo\_gmm\_multi, [2](#)  
demo\_iris, [2](#)  
demo\_normal\_performance, [3](#)  
demo\_simple\_gamma, [3](#)  
demo\_simple\_gaussian, [4](#)  
  
gmm, [4](#)  
  
KSD, [5](#)  
  
likelihoodgmm, [6](#)  
  
perturbgmm, [7](#)  
plotgmm, [8](#)  
posteriorgmm, [8](#)  
  
rgmm, [9](#)  
  
scorefunctiongmm, [10](#)