Package ‘mmfit’

February 23, 2016

Type Package

Title mmfit: fitting distributions to data and assessing model fit

Version 1.0

Date 2016-02-23

Author Guicheng Wu, Eric Sturzinger, Rafael Lourenco

Maintainer Guicheng Wu <gchwu@ucdavis.edu>

Description fitting distributions to data and assessing model fit

License GPL-2

R topics documented:

`mmfit-package` *mmfit: fitting distributions to data and assessing model fit*

Description

fitting distributions to data and assessing model fit

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.
The user mainly use three functions: plot.mmfit(), print.mmfit(), summary.mmfit().

Author(s)

Guicheng Wu, Eric Sturzinger, Rafael Lourenco

Maintainer: Guicheng Wu <gchwu@ucdavis.edu>

Examples

```
###########################################################################################################
# begin: test function

#test Poisson esitimated result
testmmfitPoisson = function() {
  poissonData <- poisson_sim(10000, 5)
  start <- c(2)
  #result <- poisson_func(start, poissonData)
  testResult <- mmfit(poisson_func, poissonData, start, 'poisson')
  return(testResult)
}

#test power law estimated result
testmmfitPowerlaw = function() {
  powerlawData = power_law_sim(10000, 1, 10)
  start <- 0.5
  #result <- power_law_func(start, powerlawData)
  testResult <- mmfit(power_law_func, powerlawData, start, 'powerlaw')
  return(testResult)
}

#test gamma estimated result
testmmfitGamma = function () {
  gammaData <- gamma_sim(10000, 7.5, 4)
  start <- c(7, 2)
  testResult <- mmfit(gamma_func, gammaData, start, 'gamma')

  return(testResult)
}

#test beta estimated result
testmmfitBeta = function() {
```

```
  betaData <- beta_sim(10000, 7.5, 4)
  start <- c(5, 2)
  testResult <- mmfit(beta_func, betaData, start, 'beta')

  return(testResult)
}

# test poisson estimated result
testmmfitPoissonMixture = function() {
  poissonMixtureData <- poisson_mixture_sim(10000, 6, 20, 0.3)
  start <- c(4, 17, 0.2)
  testResult <- mmfit(poisson_mixture_func, poissonMixtureData, start, 'poissonMixture')

  return(testResult)
}

# test plot estimated result
testmmfitExpMixture = function() {
  expMixtureData <- exp_mixture_sim(10000, 3, 10, 0.3)
  start <- c(2, 8, 0.2)
  testResult <- mmfit(exp_mixture_func, expMixtureData, start, 'expMixture')

  return(testResult)
}


#test plot poisson
testPlotPoisson = function() {
  poissonData <- poisson_sim(10000, 8)
  start <- c(1)
  plot.mmfit(poisson_func, poissonData, start, 'poisson')
}

# test power law
testPlotPowerlaw = function() {
  powerlawData <- power_law_sim(1000, 1, 3)
  start <- c(2)
  plot.mmfit(power_law_func, powerlawData, start, 'powerlaw')
}

#test plot gamma
testPlotGamma = function() {
  gammaData <- gamma_sim(1000, 2, 0.5)
  start <- c(1, 0.2)

  plot.mmfit(gamma_func, gammaData, start, 'gamma')
}

testPlotUserFunc1 = function() {
  poissonData <- poisson_sim(10000, 8)
  start <- c(lambda = 1)
  plot.mmfit(poisson_func, poissonData, start, '', poisson_density)
}

testPlotUserFun2 = function() {
  gammaData <- gamma_sim(1000, 2, 0.5)
  start <- c('shape' = 1, 'rate' = 0.2)
```

```
  plot.mmfit(gamma_func, gammaData, start, 'user', gamma_density)
}

#test plot beta
testPlotBeta = function() {
  betaData <- beta_sim(1000, 4, 3)
  start <- c(0.7, 1)

  plot.mmfit(beta_func, betaData, start, 'beta')
  #mmfit(beta_func, betaData, start, 'beta')
}

# test plot poisson mixture
testPlotPoisMixture = function() {
  mixtureData <- poisson_mixture_sim(100000, 6, 20, 0.3)
  start <- c(5, 19, 0.2)

  plot.mmfit(poisson_mixture_func, mixtureData, start, 'poissonMixture')
}

# test plot exponential mxiture
testPlotExpMixture = function() {
  expMixtureData <- exp_mixture_sim(100000, 5, 8, 0.3)
  start <- c(6, 7, 0.5)

  plot.mmfit(exp_mixture_func, expMixtureData, start, 'expMixture')
}

testPrintPoisson = function() {
  poissonData <- poisson_sim(10000, 8)
  start <- c(1)
  print.mmfit(poisson_func, poissonData, start, 'poisson')
}

testPrintPowerlaw = function() {
  powerlawData <- power_law_sim(10000, 1, 4)
  start <- c(k = 3)
  print.mmfit(power_law_func, powerlawData, start, 'powerlaw')
}

testPrintGamma = function() {
  gammaData <- gamma_sim(10000, 7.5, 4)
  start <- c(alpha = 5, beta = 2)

  print.mmfit(gamma_func, gammaData, start, 'gamma')
}

testPrintBeta = function() {
  betaData <- beta_sim(10000, 2, 4)
  start <- c(alpha = 0.5, beta = 1)

  print.mmfit(beta_func, betaData, start, 'beta')
}

testPrintPoisMixture = function() {
  mixtureData <- poisson_mixture_sim(100000, 6, 20, 0.3)
```

```
  start <- c(lambda1 = 5, lambda2 = 19, r = 0.2)

  print.mmfit(poisson_mixture_func, mixtureData, start, 'poissonMixture')
}

testPrintExpMixture = function() {
  expMixtureData <- exp_mixture_sim(100000, 0.1, 0.1, 0.3)
  start <- c(lambda1 = 0.1, lambda2 = 0.3, r = 0.4)

  print.mmfit(exp_mixture_func, expMixtureData, start, 'expMixture')
}

testSummaryPoisson = function() {
  poissonData <- poisson_sim(10000, 8)
  start <- c(1)
  summary.mmfit(poisson_func, poissonData, start, 'poisson')
}
# end: test function
#######################################################################################################
```

beta_density *Function for Beta PDF*

Description

This function takes argument shape1 and shape2 and computes a beta distribution based upon those variables. In the function we use "dbeta" which is a built in function that actually computes the function.

Usage

```
beta_density(x, shape1, shape2)
```

Arguments

x	x means to plot the values which dbeta generates on the x-axis.
shape1	Shape1 refers to the commonly known parameter alpha.
shape2	Shape2 refers to the commonly known parameter beta.

beta_func *Beta Moments Function*

Description

Function that contains the moment functions for Beta distributed random variables. Uses the gmm() function to estimate sample parameters based on sample data.

beta_density(theta, x)

\itemtheta Theta is a vector of 2 parameters, commonly known as alpha and beta, from which the population mean calculated. Alpha and Beta are the initial guess parameters from the user.

\itemxx refers to the vector of sample data, in this case a sample of beta distributed RVs.

beta_sim *Beta Data Simulation*

Description

Function that generates n beta distributed random variables.

beta_density(n, shape1, shape2)

\itemn n is the desired sample size of the user.

\itemshape1shape1 refers to the common parameter alpha that is used in generating the data.

\itemshape2 SHape2 is the common parameter beta.

expMixture_density *Moment function for a Mixture of Exponential Distributions*

Description

This function calculates the estimated parameters of simulated data asssuming it is a mixture of exponential distributions.

Usage

```
expMixture_density(x, lambda1, lambda2, r)
```

Arguments

`x`	x is the vector of simulated data or sample data.
`lambda1`	Lambda1 is the rate at which data point is generated by the first exponentially distributed RV.
`lambda2`	Lambda2 is the rate at which data point is generated by the second exponentially distributed RV.
`r`	r is the probability that the exp distributed RV is a value from system 1, where 1-r is the probability that the value belongs to system 2.

exponential *Exponential RV Generation*

Description

This function generates 1 exponentially distributed random variable with a probability r that it will be with rate1 or rate2.

Usage

```
exponential(lambda1, lambda2, r)
```

Arguments

`lambda1`	lambda1 is the rate of the first exponential distribution to generate.
`lambda2`	lambda2 is the rate of the first exponential distribution to generate.
`r`	r is the probability that a value will be generated with rate lambda1 and 1-r the the probabiliyt that value will be generated with rate lambda2.

`exp_mixture_func` *Exponential Mixture Moment Function*

Description

This function is supplied to the gmm() function for parameter computation when dealing with a mixture of exponentially distributed RVs.

Usage

```
exp_mixture_func(theta, x)
```

Arguments

`theta`	theta is a vector of the initial guesses of parameters: lambda1, lambda2, and r.
`x`	x is the vector of simulation data of a mixture of two expnontially distributed random variables.

`exp_mixture_sim` *Exponential Mixture Simulation Data*

Description

Thie function generates data of a mixture of two exponentially distributed random variables to supply to the gmm() function for computation and parameter estimation.

beta_density(n, lambda1, lambda2, r)

\itemn n is the desired sample size of the simulation data. For mixtures, we must use a very large sample size. ~> 100000

\itemlambda1Lambda1 refers to the rate of the first exponentially distributed RV.

\itemlambda2Lambda2 refers to the rate of the first exponentially distributed RV.

\itemrr refers to the probability that the RV comes from system 1, and 1-r the probability that it comes from system 2.

gamma_density *Gamma Density Function*

Description

Function that is used to plot the theoretical gamma density function with the estimateed parameters.

gamma_densityx, shape, rate

\itemx x is the sample data required to plot the function against.

\itemshapeshape is the parameter alpha used to compute density values.

\itemrate rate is the parameter alpha used to compute density values.

gamma_func *Gamma Distribution Moments Function*

Description

Function that is used to calculate 1st and second moments of gamma distribution and ffed to the gmm() function.

gamma_func(theta, x)

\itemtheta theta is the vector of initial guess parameters, alpha and beta.

\itemxx refers to the sample data used.

gamma_sim *gamma simulation function*

Description

The function will generate gamma simulated data

Usage

```
gamma_sim(n, shape, rate)
```

Arguments

n	n is the size of data
shape	shape is a gamma distribution parameter
rate	rate is a gamma distribution parameter

`mmfit` *mmfit function*

Description

The mmfit functiion will return a "mmfit" class object, which includes the estimated parameters, the standard errors and two graphs.

Usage

```
mmfit(g, x, start, distributionType, user_density_func = "")
```

Arguments

`g`	g is the moment function
`x`	x is the simulated data
`start`	start is a vector of the guess values
`distributionType`	distributionType is a string that describes the distribution type. It can be "gamma" and e.t.c.
`user_density_func`	User_density_func is the distribution's density function

`plot.mmfit` *plot.mmfit function*

Description

This function will draw two graphs, one is to check whether the simulated data fits the certain distribution, the other is to draw empirical cdf with its K-S confidence band.

Usage

```
plot.mmfit(g, x, start, distributionType, user_density_func)
```

Arguments

`g`	g is the moment function
`x`	x is the simulated data
`start`	start is a vector of the guess values
`distributionType`	distributionType is a string that describes the distribution type. It can be "gamma" and e.t.c.
`user_density_func`	User_density_func is the distribution's density function

poisMixture_density *poisson mixture density function*

Description

This function is poisson mixture density function.

Usage

```
poisMixture_density(x, lambda1, lambda2, r)
```

Arguments

`x`	x is the sample data
`lambda1`	lambda1 is the parameter of the first part of poisson mixture
`lambda2`	lambda1 is the parameter of the second part of poisson mixture
`r`	r is the parameter of proportion of poisson mixture

poisson2 *poisson mixture density function*

Usage

```
poisson2(lambda1, lambda2, r)
```

Arguments

`lambda1`	lambda1 represents the poisson parameter of the first part of poisson mixture
`lambda2`	lambda2 represents the poisson parameter of the second part of poisson mixture
`r`	r represents the porportion of the first part of poisson mixture

poisson_density *poisson density function*

Description

This function is the poisson density function

Usage

```
poisson_density(x, lambda)
```

Arguments

`x`	x is the sample data
`lambda`	lambda is the poisson parameter

poisson_func *poisson moment function*

Description

This function is the poisson moment function

Usage

```
poisson_func(theta, x)
```

Arguments

theta	theta is the vector of guess values
x	x is the sample data

poisson_mixture_func *poisson mixture moment function*

Description

This function is the moment function of poisson mixture

Usage

```
poisson_mixture_func(theta, x)
```

Arguments

theta	theta is the vector of guess value
x	x is the simulated data

poisson_mixture_sim *poission mixture simulation function*

Description

This function generates poisson mixture simulation data

Usage

```
poisson_mixture_sim(n, lambda1, lambda2, r)
```

Arguments

n	n is the size of data
lambda1	lambda1 is the parameter of the first part poisson
lambda2	lambda1 is the parameter of the second part poisson
r	r is the proportion of the first part poisson

poisson_sim *poisson data simulation function*

Description

This function will generate poisson simuated data

Usage

```
poisson_sim(n, lambda)
```

Arguments

n	n is the size of data
lambda	lambda is the parameter of poisson

powerlaw_density *Power law density*

Description

This function define the power density

Usage

```
powerlaw_density(x, theta)
```

Arguments

x	x is the sample data
theta	theta is the estimated parameter for the power law density

power_law_func *power law moment function*

Description

This is the power law moment function

Usage

```
power_law_func(theta, x)
```

Arguments

theta	theta is the initial guess values of the parameters
x	x is the sample data

`power_law_sim` *power law simualtion function*

Description

This function generates the power law simulated data

Usage

```
power_law_sim(n, xmin, k)
```

Arguments

`n`	n is the size of simulated data
`xmin`	xmin is the lower bound
`k`	k is the estimated parameter for power law

`print.mmfit` *print.mmfit function*

Description

print.mmfit function will print the estimated parameters and standard errors.

Usage

```
print.mmfit(g, x, start, distributionType, user_density_func)
```

Arguments

`g`	g is the moment function
`x`	x is the simulated data
`start`	start is a vector of the guess values
`distributionType`	distributionType is a string that describes the distribution type. It can be "gamma" and e.t.c.
`user_density_func`	User_density_func is the distribution's density function

`summary.mmfit` *summary.mmfit function*

Description

This function will summary everything the mmfit method returned, including the estimated parameters, the standard errors, and the graphs.

Usage

```
summary.mmfit(g, x, start, distributionType, user_density_func)
```

Arguments

`g`	g is the moment function
`x`	x is the simulated data
`start`	start is a vector of the guess values
`distributionType`	distributionType is a string that describes the distribution type. It can be "gamma" and e.t.c.
`user_density_func`	User_density_func is the distribution's density function

Index