

Package ‘visualize’

July 2, 2014

Type Package

Title Graph Probability Distributions with User Supplied Parameters and Stats.

Version 4.2

Date 2013-08-03

Author James Balamuta

Maintainer James Balamuta <james.balamuta@gmail.com>

Description Graphs the pdf or pmf and highlights what area or probability is present in user defined locations. Visualize is able to provide lower tail, bounded, upper tail, and two tail calculations. Supports strict and equal to inequalities. Also provided on the graph is the mean and variance of the distribution.

License MIT License

URL <http://thecoatlessprofessor.com>

NeedsCompilation no

Repository CRAN

Date/Publication 2013-08-03 16:59:15

R topics documented:

visualize-package	2
visualize.beta	3
visualize.binom	4
visualize.cauchy	5
visualize.chisq	6
visualize.continuous	7
visualize.discrete	8
visualize.distributions	9
visualize.exp	10
visualize.f	11

visualize.gamma	12
visualize.geom	13
visualize.hyper	14
visualize.it	15
visualize.lnorm	17
visualize.logis	18
visualize.nbinom	19
visualize.norm	20
visualize.pois	20
visualize.t	21
visualize.unif	22
visualize.wilcox	23

Index	25
--------------	-----------

visualize-package	<i>Graph Probability Distributions with User Supplied Parameters and Stats.</i>
-------------------	---

Description

Graphs the pdf or pmf and highlights what area or probability is present in user defined locations. Visualize is able to provide lower tail, bounded, two tail, and upper tail calculations. Supports strict and equal to inequalities. Also provided on the graph is the mean and variance of the distribution.

Details

Package: visualize
 Type: Package
 Version: 4.2
 Date: 2013-08-03
 License: MIT License

Author(s)

James Balamuta
 Maintainer: James Balamuta <james.balamuta@gmail.com>

Examples

```
##visualize.it acts as the general wrapper.
##For guided application of visualize, see the visualize.distr_name list.
#Binomial distribution evaluated at lower tail.
visualize.it(dist = 'binom', stat = 2, params = list(size = 4,prob = .5), section = "lower",
```

```

        strict = TRUE)
visualize.binom(stat = 2, size = 4, prob = .5, section = "lower", strict = TRUE)

#Set to shade inbetween a bounded region.
visualize.it(dist = 'norm', stat = c(-1,1), list(mu=0,sd=1), section="bounded")
visualize.norm(stat = c(-1,1), mu = 0,sd = 1, section = "bounded")

#Gamma distribution evaluated at upper tail.
visualize.it(dist = 'gamma', stat = 2, params = list(alpha = 2, theta = 1),section="upper")
visualize.gamma(stat = 2, alpha = 2, theta = 1, section="upper")

```

visualize.beta	<i>Visualize Beta Distribution</i>
----------------	------------------------------------

Description

Generates a plot of the Beta distribution with user specified parameters.

Usage

```
visualize.beta(stat = 1, alpha = 3, beta = 2, section = "lower")
```

Arguments

stat	a statistic to obtain the probability from. When using the "bounded" condition, you must supply the parameter as <code>stat = c(lower_bound, upper_bound)</code> . Otherwise, a simple <code>stat = desired_point</code> will suffice.
alpha	alpha is considered to be <i>shape1</i> by R's implementation of the beta distribution. alpha must be greater than 0.
beta	beta is considered to be <i>shape2</i> by R's implementation of the beta distribution. beta must be greater than 0.
section	Select how you want the statistic(s) evaluated via <code>section=</code> either "lower", "bounded", "upper", or "tails".

Value

Returns a plot of the distribution according to the conditions supplied.

Author(s)

James Balamuta

See Also

[visualize.it](#), [dbeta](#).

Examples

```
#Evaluates lower tail.
visualize.beta(stat = 1, alpha = 2, beta = 3, section = "lower")

#Evaluates bounded region.
visualize.beta(stat = c(.5,1), alpha = 4, beta = 3, section = "bounded")

#Evaluates upper tail.
visualize.beta(stat = 1, alpha = 2, beta = 3, section = "upper")
```

visualize.binom	<i>Visualize Binomial Distribution</i>
-----------------	--

Description

Generates a plot of the Binomial distribution with user specified parameters.

Usage

```
visualize.binom(stat = 1, size = 3, prob = 0.5, section = "lower", strict = FALSE)
```

Arguments

stat	a statistic to obtain the probability from. When using the "bounded" condition, you must supply the parameter as <code>stat = c(lower_bound, upper_bound)</code> . Otherwise, a simple <code>stat = desired_point</code> will suffice.
size	size of sample.
prob	probability of picking object.
section	Select how you want the statistic(s) evaluated via <code>section=</code> either "lower", "bounded", "upper", or "tails".
strict	Determines whether the probability will be generated as a strict (<, >) or equal to (<=, >=) inequality. <code>strict=</code> requires either values = 0 or =FALSE for equal to OR values =1 or =TRUE for strict. For bounded condition use: <code>strict=c(0, 1)</code> or <code>strict=c(FALSE, TRUE)</code> .

Author(s)

James Balamuta

See Also

[visualize.it](#), [dbinom](#).

Examples

```
#Evaluates lower tail with equal to inequality.
visualize.binom(stat = 1, size = 3, prob = 0.5, section = "lower", strict = FALSE)

#Evaluates bounded region with lower bound equal to and upper bound strict inequality.
visualize.binom(stat = c(1,2), size = 5, prob = 0.35, section = "bounded", strict = c(0,1))

#Evaluates upper tail with strict inequality.
visualize.binom(stat = 1, size = 3, prob = 0.5, section = "upper", strict = TRUE)
```

visualize.cauchy *Visualize Cauchy Distribution*

Description

Generates a plot of the Cauchy distribution with user specified parameters.

Usage

```
visualize.cauchy(stat = 1, location = 2, scale = 1, section = "lower")
```

Arguments

stat	a statistic to obtain the probability from. When using the "bounded" condition, you must supply the parameter as <code>stat = c(lower_bound, upper_bound)</code> . Otherwise, a simple <code>stat = desired_point</code> will suffice.
location	location parameter
scale	scale parameter
section	Select how you want the statistic(s) evaluated via <code>section=</code> either "lower", "bounded", "upper", or "tails".

Value

Returns a plot of the distribution according to the conditions supplied.

Author(s)

James Balamuta

See Also

[visualize.it](#), [dcauchy](#).

Examples

```
#Evaluates lower tail.
visualize.cauchy(stat = 1, location = 4, scale = 2, section = "lower")

#Evaluates bounded region.
visualize.cauchy(stat = c(3,5), location = 5, scale = 3, section = "bounded")

#Evaluates upper tail.
visualize.cauchy(stat = 1, location = 4, scale = 2, section = "upper")
```

visualize.chisq	<i>Visualize Chi-squared Distribution</i>
-----------------	---

Description

Generates a plot of the Chi-squared distribution with user specified parameters.

Usage

```
visualize.chisq(stat = 1, df = 3, section = "lower")
```

Arguments

stat	a statistic to obtain the probability from. When using the "bounded" condition, you must supply the parameter as <code>stat = c(lower_bound, upper_bound)</code> . Otherwise, a simple <code>stat = desired_point</code> will suffice.
df	degrees of freedom of Chi-squared distribution.
section	Select how you want the statistic(s) evaluated via <code>section=</code> either "lower", "bounded", "upper", or "tails".

Value

Returns a plot of the distribution according to the conditions supplied.

Author(s)

James Balamuta

See Also

[visualize.it](#), [dchisq](#).

Examples

```
#Evaluates lower tail.
visualize.chisq(stat = 1, df = 3, section = "lower")
#Evaluates bounded region.
visualize.chisq(stat = c(1,2), df = 6, section = "bounded")
#Evaluates upper tail.
visualize.chisq(stat = 1, df = 3, section = "upper")
```

visualize.continuous *Graphing function for Continuous Distributions.*

Description

Handles how continuous distributions are graphed. Users should not use this function. Instead, users should use `link{visualize.it}`.

Usage

```
visualize.continuous(dist, stat = c(0, 1), params, section = "lower")
```

Arguments

<code>dist</code>	contains the distribution from <code>link{visualize.distributions}</code> .
<code>stat</code>	a statistic to obtain the probability from. When using the "bounded" condition, you must supply the parameter as <code>stat = c(lower_bound, upper_bound)</code> . Otherwise, a simple <code>stat = desired_point</code> will suffice.
<code>params</code>	A list that must contain the necessary parameters for each distribution. For example, <code>params = list(mu = 1, sd = 1)</code> would be for a normal distribution with mean 1 and standard deviation 1. If you are not aware of the parameters for the distribution, consider using the <code>visualize.dist_name</code> functions listed under the "See Also" section.
<code>section</code>	Select how you want the statistic(s) evaluated via <code>section=</code> either "lower", "bounded", "upper", or "tails".

Author(s)

James Balamuta

See Also

[visualize.it](#), [visualize.beta](#), [visualize.chisq](#), [visualize.exp](#), [visualize.gamma](#), [visualize.norm](#), [visualize.unif](#), [visualize.cauchy*](#), [visualize.f*](#), [visualize.lnorm*](#), [visualize.t*](#), [visualize.wilcox*](#), [visualize.logis*](#).

* = added in v2.0.

Examples

```
#Function does not have dist look up, must go through visualize.it
visualize.it(dist='norm', stat = c(0,1), params = list(mu = 1, sd = 1), section = "bounded")
```

visualize.discrete *Graphing function for Discrete Distributions.*

Description

Handles how discrete distributions are graphed. Users should not use this function. Instead, users should use `link{visualize.it}`.

Usage

```
visualize.discrete(dist, stat = c(0, 1), params, section = "lower", strict)
```

Arguments

dist	contains the distribution from <code>link{visualize.distributions}</code> .
stat	a statistic to obtain the probability from. When using the "bounded" condition, you must supply the parameter as <code>stat = c(lower_bound, upper_bound)</code> . Otherwise, a simple <code>stat = desired_point</code> will suffice.
params	A list that must contain the necessary parameters for each distribution. For example, <code>params = list(n = 5, prob = .25)</code> would be for a binomial distribution with size 5 and probability .25. If you are not aware of the parameters for the distribution, consider using the <code>visualize.dist_name</code> functions listed under the "See Also" section.
section	Select how you want the statistic(s) evaluated via <code>section=</code> either "lower", "bounded", "upper", or "tails".
strict	Determines whether the probability will be generated as a strict (<, >) or equal to (<=, >=) inequality. <code>strict=</code> requires either values = 0 or =FALSE for equal to OR values =1 or =TRUE for strict. For bounded condition use: <code>strict=c(0, 1)</code> or <code>strict=c(FALSE, TRUE)</code> .

Author(s)

James Balamuta

See Also

[visualize.it](#), [visualize.binom](#), [visualize.geom](#), [visualize.hyper](#), [visualize.nbinom](#), [visualize.pois](#).

Examples

```
#Function does not have dist look up, must go through visualize.it
visualize.it(dist='geom', stat = c(2,4), params = list(prob = .75), section = "bounded",
            strict = c(0,1))
```

 visualize.distributions

Visualize's Supported Distributions

Description

All of visualize's supported distributions with their density, probability, and quantile functions. In addition, mean and variance functions are present. Other descriptors also exist and are documented below.

Format

Distributions are loaded with the following format:

type:	specify either "continuous" or "discrete" to direct the query to the right graph handler.
name:	specify the name of the distribution. In example, "Poisson Distribution." This is used in the main graph title.
variable:	specify the variable in probability statement. In example, $P(z < 5)$. This is used in the probability subtitle.
varsymbols:	specify the variable symbols for distribution. In example, $\mu = 1$ $sd = 2$. This is used in the distribution subtitle.
params:	specify the amount of params required for distribution. This is used in the first error handling check to ensure the correct number of params is supplied.
init(params, ...):	Function that generates the mean and variance of a distribution.
density(x, params, ncp = 0, lower.tail = TRUE, log = FALSE, ...):	Function that provides the density value using vectors of quantiles from the distribution. This serves as a wrapper for <code>ddistr_name</code> .
probability(x, params, ncp = 0, lower.tail = TRUE, log.p = FALSE, ...)	Function that provides the probability value using vectors of quantiles from the distribution. This serves as a wrapper for <code>pdistr_name</code> .
quantile(x, params, ncp = 0, lower.tail = TRUE, log.p = FALSE, ...)	Function that provides the quantile value using vectors of probabilities from the distribution. This serves as a wrapper for <code>qdistr_name</code> .

The distributions currently available to use are:

Distribution	r Name	Distribution	r Name
Beta	beta	Lognormal*	lnorm
Binomial	binom	Negative Binomial	nbinom
Cauchy*	cauchy	Normal	norm
Chisquare	chisq	Poisson	pois

Exponential	exp	Student t*	t
F*	f	Uniform	unif
Gamma	gamma	Geometric	geom
Hypergeometric	hyper	Wilcoxon*	wilcox
Logistic*	logis		

* denotes the distribution was added in v2.0.

Author(s)

James Balamuta

Examples

```
visualize.distributions = list(
  'beta' = list(
    type = "continuous",
    name = "Beta Distribution",
    variable = "b",
    varsymbols = c("\u03B1", "\u03B2"),
    params = 2,
    init = function(params, ...) {
      shape1 = params[[1]]; shape2 = params[[2]]
      if(shape1 <= 0 || shape2 <= 0) stop("Error: Need alpha, beta > 0")
      mean = shape1 / (shape1 + shape2)
      var = (shape1 * shape2) / ((shape1 + shape2 + 1) * (shape1 + shape2)^2)
      c(mean, var)
    },
    density = function(x, params, ncp = 0, lower.tail = TRUE, log = FALSE, ...){
      if(params[[1]] <= 0 || params[[2]] <= 0) stop("Error: Need alpha, beta > 0")
      dbeta(x, params[[1]], params[[2]], ncp = ncp, log = log)
    },
    probability = function(q, params, ncp = 0, lower.tail = TRUE, log.p = FALSE, ...){
      if(params[[1]] <= 0 || params[[2]] <= 0) stop("Error: Need alpha, beta > 0")
      pbeta(q, params[[1]], params[[2]], ncp = ncp, lower.tail = lower.tail, log.p = log.p)
    },
    quantile = function(p, params, ncp = 0, lower.tail = TRUE, log.p = FALSE, ...){
      if(params[[1]] <= 0 || params[[2]] <= 0) stop("Error: Need alpha, beta > 0")
      qbeta(p, params[[1]], params[[2]], ncp = ncp, lower.tail = lower.tail, log.p = log.p)
    }
  )
)
```

visualize.exp

Visualize Exponential Distribution

Description

Generates a plot of the Exponential distribution with user specified parameters.

Usage

```
visualize.exp(stat = 1, theta = 1, section = "lower")
```

Arguments

stat	a statistic to obtain the probability from. When using the "bounded" condition, you must supply the parameter as <code>stat = c(lower_bound, upper_bound)</code> . Otherwise, a simple <code>stat = desired_point</code> will suffice.
theta	vector of rates
section	Select how you want the statistic(s) evaluated via <code>section=</code> either "lower", "bounded", "upper", or "tails".

Value

Returns a plot of the distribution according to the conditions supplied.

Author(s)

James Balamuta

See Also

[visualize.it](#), [dexp](#).

Examples

```
#Evaluates lower tail.
visualize.exp(stat = .5, theta = 3, section = "lower")
#Evaluates bounded region.
visualize.exp(stat = c(1,2), theta = 3, section = "bounded")
#Evaluates upper tail.
visualize.exp(stat = .5, theta = 3, section = "upper")
```

visualize.f

Visualize F distribution

Description

Generates a plot of the F distribution with user specified parameters.

Usage

```
visualize.f(stat = 1, df1 = 5, df2 = 4, section = "lower")
```

Arguments

stat	a statistic to obtain the probability from. When using the "bounded" condition, you must supply the parameter as <code>stat = c(lower_bound, upper_bound)</code> . Otherwise, a simple <code>stat = desired_point</code> will suffice.
df1	First Degrees of Freedom
df2	Second Degrees of Freedom
section	Select how you want the statistic(s) evaluated via <code>section=</code> either "lower", "bounded", "upper", or "tails".

Value

Returns a plot of the distribution according to the conditions supplied.

Author(s)

James Balamuta

See Also

[visualize.it](#), [df](#).

Examples

```
#Evaluates lower tail.
visualize.f(stat = 1, df1 = 5, df2 = 4, section = "lower")

#Evaluates bounded region.
visualize.f(stat = c(3,5), df1 = 6, df2 = 3, section = "bounded")

#Evaluates upper tail.
visualize.f(stat = 1, df1 = 5, df2 = 4, section = "upper")
```

visualize.gamma

Visualize Gamma Distribution

Description

Generates a plot of the Gamma distribution with user specified parameters.

Usage

```
visualize.gamma(stat = 1, alpha = 1, theta = 1, section = "lower")
```

Arguments

stat	a statistic to obtain the probability from. When using the "bounded" condition, you must supply the parameter as <code>stat = c(lower_bound, upper_bound)</code> . Otherwise, a simple <code>stat = desired_point</code> will suffice.
alpha	alpha is considered to be <i>shape</i> by R's implementation of the gamma distribution. alpha must be greater than 0.
theta	theta is considered to be <i>rate</i> by R's implementation of the gamma distribution. theta must be greater than 0.
section	Select how you want the statistic(s) evaluated via <code>section=</code> either "lower", "bounded", "upper", or "tails".

Author(s)

James Balamuta

See Also

[visualize.it](#), [dgamma](#).

Examples

```
#Evaluate lower tail.
visualize.gamma(stat = 1, alpha = 3, theta = 1, section = "lower")

#Evaluate bounded section.
visualize.gamma(stat = c(0.75,1), alpha = 3, theta = 1, section = "bounded")

#Evaluate upper tail.
visualize.gamma(stat = 1, alpha = 3, theta = 1, section = "upper")
```

visualize.geom

Visualize Geometric Distribution

Description

Generates a plot of the Geometric distribution with user specified parameters.

Usage

```
visualize.geom(stat = 1, prob = 0.3, section = "lower", strict = FALSE)
```

Arguments

stat	a statistic to obtain the probability from. When using the "bounded" condition, you must supply the parameter as <code>stat = c(lower_bound, upper_bound)</code> . Otherwise, a simple <code>stat = desired_point</code> will suffice.
prob	probability of picking object.
section	Select how you want the statistic(s) evaluated via <code>section=</code> either "lower", "bounded", "upper", or "tails".
strict	Determines whether the probability will be generated as a strict (<, >) or equal to (<=, >=) inequality. <code>strict=</code> requires either values = 0 or =FALSE for equal to OR values =1 or =TRUE for strict. For bounded condition use: <code>strict=c(0,1)</code> or <code>strict=c(FALSE, TRUE)</code> .

Author(s)

James Balamuta

See Also

[visualize.it](#), [dgeom](#).

Examples

```
#Evaluates lower tail.
visualize.geom(stat = 1, prob = 0.5, section = "lower", strict = FALSE)

#Evaluates bounded region.
visualize.geom(stat = c(1,3), prob = 0.35, section = "bounded", strict = c(0,1))

#Evaluates upper tail.
visualize.geom(stat = 1, prob = 0.5, section = "upper", strict = 1)
```

visualize.hyper

Visualize Hypergeometric Distribution

Description

Generates a plot of the Hypergeometric distribution with user specified parameters.

Usage

```
visualize.hyper(stat = 1, m = 5, n = 4, k = 2, section = "lower", strict = FALSE)
```

Arguments

stat	a statistic to obtain the probability from. When using the "bounded" condition, you must supply the parameter as <code>stat = c(lower_bound, upper_bound)</code> . Otherwise, a simple <code>stat = desired_point</code> will suffice.
m	m white balls. m must be greater than 0.
n	n black balls. n must be greater than 0.
k	draw k balls without replacement.
section	Select how you want the statistic(s) evaluated via <code>section=</code> either "lower", "bounded", "upper", or "tails".
strict	Determines whether the probability will be generated as a strict (<, >) or equal to (<=, >=) inequality. <code>strict=</code> requires either values = 0 or =FALSE for equal to OR values =1 or =TRUE for strict. For bounded condition use: <code>strict=c(0, 1)</code> or <code>strict=c(FALSE, TRUE)</code> .

Author(s)

James Balamuta

See Also[visualize.it](#), [dhyper](#).**Examples**

```
#Evaluates lower tail.
visualize.hyper(stat = 1, m=4, n=5, k=3, section = "lower", strict = 0)

#Evaluates bounded region.
visualize.hyper(stat = c(2,4), m=14, n=5, k=2, section = "bounded", strict = c(0,1))

#Evaluates upper tail.
visualize.hyper(stat = 1, m=4, n=5, k=3, section = "upper", strict = 1)
```

`visualize.it`*Visualize's Processing Function*

Description

Acts as a director of traffic and first line of error handling regarding submitted visualization requests. This function should only be used by advanced users.

Usage

```
visualize.it(dist = "norm", stat = c(0, 1), params = list(mu = 0, sd = 1),
            section = "lower", strict = c(0,1))
```

Arguments

<code>dist</code>	a string that should contain a supported probability distributions name in R. Supported continuous distributions: "beta", "chisq", "exp", "gamma", "norm", and "unif". Supported discrete distributions: "binom", "geom", "hyper", "nbinom", and "pois".
<code>stat</code>	a statistic to obtain the probability from. When using the "bounded" condition, you must supply the parameter as <code>stat = c(lower_bound, upper_bound)</code> . Otherwise, a simple <code>stat = desired_point</code> will suffice.
<code>params</code>	A list that must contain the necessary parameters for each distribution. For example, <code>params = list(mu = 1, sd = 1)</code> would be for a normal distribution with mean 1 and standard deviation 1. If you are not aware of the parameters for the distribution, consider using the <code>visualize.dist</code> functions listed under the "See Also" section.
<code>section</code>	Select how you want the statistic(s) evaluated via <code>section=</code> either "lower", "bounded", "upper", or "tails".
<code>strict</code>	Determines whether the probability will be generated as a strict (<, >) or equal to (<=, >=) inequality. <code>strict=</code> requires either values = 0 or =FALSE for strict OR values =1 or =TRUE for equal to. For bounded condition use: <code>strict=c(0, 1)</code> or <code>strict=c(FALSE, TRUE)</code> .

Value

Returns a plot of the distribution according to the conditions supplied.

Author(s)

James Balamuta

References

<http://cran.r-project.org/web/views/Distributions.html>

See Also

[visualize.beta](#), [visualize.chisq](#), [visualize.exp](#), [visualize.gamma](#), [visualize.norm](#), [visualize.unif](#), [visualize.binom](#), [visualize.geom](#), [visualize.hyper](#), [visualize.nbinom](#), [visualize.pois](#).

Examples

```
# Defaults to lower tail evaluation
visualize.it(dist = 'norm', stat = 1, list(mu = 3 , sd = 2), section = "lower")

#Set to evaluate the upper tail.
visualize.it(dist = 'norm', stat = 1, list(mu=3,sd=2),section="upper")

#Set to shade inbetween a bounded region.
visualize.it(dist = 'norm', stat = c(-1,1), list(mu=0,sd=1), section="bounded")

#Gamma distribution evaluated at upper tail.
```



```
visualize.it(dist = 'gamma', stat = 2, params = list(alpha=2,beta=1),section="upper")

#Binomial distribution evaluated at lower tail.
visualize.it('binom', stat = 2, params = list(n=4,p=.5))
```

visualize.lnorm *Visualize Log Normal Distribution*

Description

Generates a plot of the Log Normal distribution with user specified parameters.

Usage

```
visualize.lnorm(stat = 1, meanlog = 3, sdlog = 1, section = "lower")
```

Arguments

stat	a statistic to obtain the probability from. When using the "bounded" condition, you must supply the parameter as stat = c(lower_bound, upper_bound). Otherwise, a simple stat = desired_point will suffice.
meanlog	Mean of the distribution
sdlog	Standard deviation of the distribution
section	Select how you want the statistic(s) evaluated via section= either "lower", "bounded", "upper", or "tails".

Value

Returns a plot of the distribution according to the conditions supplied.

Author(s)

James Balamuta

See Also

[visualize.it](#), [dlnorm](#).

Examples

```
#Evaluates lower tail.
visualize.lnorm(stat = 1, meanlog = 3, sdlog = 1, section = "lower")

#Evaluates bounded region.
visualize.lnorm(stat = c(3,5), meanlog = 3, sdlog = 3, section = "bounded")

#Evaluates upper tail.
visualize.lnorm(stat = 1, meanlog = 3, sdlog = 1, section = "upper")
```

visualize.logis *Visualize Logistic distribution*

Description

Generates a plot of the Logistic distribution with user specified parameters.

Usage

```
visualize.logis(stat = 1, location = 3, scale = 1, section = "lower")
```

Arguments

stat	a statistic to obtain the probability from. When using the "bounded" condition, you must supply the parameter as <code>stat = c(lower_bound, upper_bound)</code> . Otherwise, a simple <code>stat = desired_point</code> will suffice.
location	Location of the distribution.
scale	Scale of the distribution.
section	Select how you want the statistic(s) evaluated via <code>section=</code> either "lower", "bounded", "upper", or "tails".

Value

Returns a plot of the distribution according to the conditions supplied.

Author(s)

James Balamuta

See Also

[visualize.it](#), [dlogis](#).

Examples

```
#Evaluates lower tail.
visualize.logis(stat = 1, location = 4, scale = 2, section = "lower")

#Evaluates bounded region.
visualize.logis(stat = c(3,5), location = 4, scale = 2, section = "bounded")

#Evaluates upper tail.
visualize.logis(stat = 1, location = 4, scale = 2, section = "upper")
```

visualize.nbinom *Visualize Negative Binomial Distribution*

Description

Generates a plot of the Negative Binomial distribution with user specified parameters.

Usage

```
visualize.nbinom(stat = 1, size = 6, prob = 0.5, section = "lower", strict = FALSE)
```

Arguments

stat	a statistic to obtain the probability from. When using the "bounded" condition, you must supply the parameter as <code>stat = c(lower_bound, upper_bound)</code> . Otherwise, a simple <code>stat = desired_point</code> will suffice.
size	number of objects.
prob	probability of picking object.
section	Select how you want the statistic(s) evaluated via <code>section=</code> either "lower", "bounded", "upper", or "tails".
strict	Determines whether the probability will be generated as a strict ($<$, $>$) or equal to ($<=$, $>=$) inequality. <code>strict=</code> requires either values = 0 or = FALSE for equal to OR values = 1 or = TRUE for strict. For bounded condition use: <code>strict=c(0, 1)</code> or <code>strict=c(FALSE, TRUE)</code> .

Author(s)

James Balamuta

See Also

[visualize.it](#), [dnbinom](#).

Examples

```
#Evaluates lower tail.
visualize.nbinom(stat = 1, size = 5, prob = 0.5, section = "lower", strict = 0)

#Evaluates bounded region.
visualize.nbinom(stat = c(1,3), size = 10, prob = 0.35, section = "bounded",
                 strict = c(TRUE, FALSE))

#Evaluates upper tail.
visualize.nbinom(stat = 1, size = 5, prob = 0.5, section = "upper", strict = 1)
```

visualize.norm *Visualize Normal Distribution*

Description

Generates a plot of the Normal distribution with user specified parameters.

Usage

```
visualize.norm(stat = 1, mu = 0, sd = 1, section = "lower")
```

Arguments

stat	a statistic to obtain the probability from. When using the "bounded" condition, you must supply the parameter as <code>stat = c(lower_bound, upper_bound)</code> . Otherwise, a simple <code>stat = desired_point</code> will suffice.
mu	mean of the Normal Distribution.
sd	standard deviation of the Normal Distribution.
section	Select how you want the statistic(s) evaluated via <code>section=</code> either "lower", "bounded", "upper", or "tails".

See Also

[visualize.it](#), [dnorm](#).

Examples

```
#Evaluates lower tail.
visualize.norm(stat = 1, mu = 4, sd = 5, section = "lower")

#Evaluates bounded region.
visualize.norm(stat = c(3,6), mu = 5, sd = 3, section = "bounded")

#Evaluates upper tail.
visualize.norm(stat = 1, mu = 3, sd = 2, section = "upper")
```

visualize.pois *Visualize Poisson Distribution*

Description

Generates a plot of the Poisson distribution with user specified parameters.

Usage

```
visualize.pois(stat = 1, lambda = 3.5, section = "lower", strict = FALSE)
```

Arguments

stat	a statistic to obtain the probability from. When using the "bounded" condition, you must supply the parameter as <code>stat = c(lower_bound, upper_bound)</code> . Otherwise, a simple <code>stat = desired_point</code> will suffice.
lambda	lambda value of the Poisson Distribution.
section	Select how you want the statistic(s) evaluated via <code>section=</code> either "lower", "bounded", "upper", or "tails".
strict	Determines whether the probability will be generated as a strict (<, >) or equal to (<=, >=) inequality. <code>strict=</code> requires either values = 0 or =FALSE for equal to OR values =1 or =TRUE for strict. For bounded condition use: <code>strict=c(0,1)</code> or <code>strict=c(FALSE, TRUE)</code> .

Author(s)

James Balamuta

See Also

[visualize.it](#), [dpois](#).

Examples

```
#Evaluates lower tail.
visualize.pois(stat = 1, lambda = 2, section = "lower", strict = FALSE)

#Evaluates bounded region.
visualize.pois(stat = c(1,3), lambda = 3, section = "bounded", strict = c(0,1))

#Evaluates upper tail.
visualize.pois(stat = 1, lambda = 2, section = "upper", strict = 1)
```

visualize.t

Visualize Student's t distribution

Description

Generates a plot of the Student's t distribution with user specified parameters.

Usage

```
visualize.t(stat = 1, df = 3, section = "lower")
```

Arguments

stat	a statistic to obtain the probability from. When using the "bounded" condition, you must supply the parameter as <code>stat = c(lower_bound, upper_bound)</code> . Otherwise, a simple <code>stat = desired_point</code> will suffice.
df	Degrees of freedom
section	Select how you want the statistic(s) evaluated via <code>section=</code> either "lower", "bounded", "upper", or "tails".

Value

Returns a plot of the distribution according to the conditions supplied.

Author(s)

James Balamuta

See Also

[visualize.it, dt.](#)

Examples

```
#Evaluates lower tail.
visualize.t(stat = 1, df = 4, section = "lower")

#Evaluates bounded region.
visualize.t(stat = c(3,5), df = 6, section = "bounded")

#Evaluates upper tail.
visualize.t(stat = 1, df = 4, section = "upper")
```

visualize.unif

Visualize Uniform Distribution

Description

Generates a plot of the Uniform distribution with user specified parameters.

Usage

```
visualize.unif(stat = 1, a = 0, b = 1, section = "lower")
```

Arguments

stat	a statistic to obtain the probability from. When using the "bounded" condition, you must supply the parameter as <code>stat = c(lower_bound, upper_bound)</code> . Otherwise, a simple <code>stat = desired_point</code> will suffice.
a	starting point. Note: $a < b$
b	end point. Note: $b > a$
section	Select how you want the statistic(s) evaluated via <code>section=</code> either "lower", "bounded", "upper", or "tails".

Author(s)

James Balamuta

See Also

[visualize.it](#), [dunif](#).

Examples

```
#Evaluates lower tail.
visualize.unif(stat = 8.75, a = 7, b = 10, section = "lower")

#Evaluates bounded region.
visualize.unif(stat = c(3,6), a = 1, b = 7, section = "bounded")

#Evaluates upper tail.
visualize.unif(stat = 2, a = 1, b = 5, section = "upper")
```

visualize.wilcox *Visualize Cauchy Distribution*

Description

Generates a plot of the Wilcoxon Rank Sum distribution with user specified parameters.

Usage

```
visualize.wilcox(stat = 1, m = 7, n = 3, section = "lower")
```

Arguments

stat	a statistic to obtain the probability from. When using the "bounded" condition, you must supply the parameter as <code>stat = c(lower_bound, upper_bound)</code> . Otherwise, a simple <code>stat = desired_point</code> will suffice.
m	Sample size from group 1.
n	Sample size from group 2.
section	Select how you want the statistic(s) evaluated via <code>section=</code> either "lower", "bounded", "upper", or "tails".

Value

Returns a plot of the distribution according to the conditions supplied.

Author(s)

James Balamuta

See Also

[visualize.it](#), [dwilcox](#).

Examples

```
#Evaluates lower tail.
visualize.wilcox(stat = 1, m = 7, n = 3, section = "lower")

#Evaluates bounded region.
visualize.wilcox(stat = c(2,3), m = 5, n = 4, section = "bounded")

#Evaluates upper tail.
visualize.wilcox(stat = 1, m = 7, n = 3, section = "upper")
```


Index

- *Topic **datasets**
 - visualize.distributions, 9
- *Topic **package**
 - visualize-package, 2
- *Topic **visualize**
 - visualize.beta, 3
 - visualize.binom, 4
 - visualize.cauchy, 5
 - visualize.chisq, 6
 - visualize.continuous, 7
 - visualize.discrete, 8
 - visualize.exp, 10
 - visualize.f, 11
 - visualize.gamma, 12
 - visualize.geom, 13
 - visualize.hyper, 14
 - visualize.it, 15
 - visualize.lnorm, 17
 - visualize.logis, 18
 - visualize.nbinom, 19
 - visualize.norm, 20
 - visualize.pois, 20
 - visualize.t, 21
 - visualize.unif, 22
 - visualize.wilcox, 23
- dbeta, 3
- dbinom, 4
- dcauchy, 5
- dchisq, 6
- dexp, 11
- df, 12
- dgamma, 13
- dgeom, 14
- dhyper, 15
- dlnorm, 17
- dlogis, 18
- dnbinom, 19
- dnorm, 20
- dpois, 21
- dt, 22
- dunif, 23
- dwilcox, 24
- visualize (visualize-package), 2
- visualize-package, 2
- visualize.beta, 3, 7, 16
- visualize.binom, 4, 8, 16
- visualize.cauchy, 5, 7
- visualize.chisq, 6, 7, 16
- visualize.continuous, 7
- visualize.discrete, 8
- visualize.distributions, 9
- visualize.exp, 7, 10, 16
- visualize.f, 7, 11
- visualize.gamma, 7, 12, 16
- visualize.geom, 8, 13, 16
- visualize.hyper, 8, 14, 16
- visualize.it, 3–8, 11–15, 15, 17–24
- visualize.lnorm, 7, 17
- visualize.logis, 7, 18
- visualize.nbinom, 8, 16, 19
- visualize.norm, 7, 16, 20
- visualize.pois, 8, 16, 20
- visualize.t, 7, 21
- visualize.unif, 7, 16, 22
- visualize.wilcox, 7, 23