Package ‘LearningStats’

April 21, 2021

Type Package

Title Elemental Descriptive and Inferential Statistics

Version 0.1.0

Description Provides tools to teach students elemental statistics. The main topics covered are descriptive statistics, probability models (discrete and continuous variables) and statistical inference (confidence intervals and hypothesis tests). One of the main advantages of this package is that allows the user to read quite a variety of types of data files with one unique command. Moreover it includes shortcuts to simple but up-to-now not in R descriptive features such a complete frequency table or an histogram with the optimal number of intervals. Related to model distributions (both discrete and continuous), the package allows the student to easy plot the mass/density function, distribution function and quantile function just detailing as input arguments the known population parameters. The inference related tools are basically confidence interval and hypothesis testing. Having defined independent commands for these two tools makes it easier for the student to understand what the software is performing, and it also helps the student to have a better knowledge on which specific tool they need to use in each situation. Moreover, the hypothesis testing commands provide not only the numeric result on the screen but also a very intuitive graph (which includes the statistic distribution, the observed value of the statistic, the rejection area and the p-value) that is very useful for the student to visualise the process. The regression section includes up to now, a simple linear model, with one single command the student can obtain the numeric summary as well as the corresponding diagram with the adjusted regression model and a legend with basic information (formula of the adjusted model and R-squared).

Language en-GB

License GPL-2

Encoding UTF-8

Depends R (>= 3.5.0)

LazyData true

RoxygenNote 7.1.1

Imports stats, grDevices, graphics, utils, tools, data.table, readxl, haven, readODS

NeedsCompilation no

Author María Isabel Borrajo-García [aut, cre],
Mercedes Conde-Amboage [aut],
Alejandra López-Pérez [aut]

Maintainer María Isabel Borrajo-García <mariaisabel.borrajo@usc.es>

Repository CRAN

Date/Publication 2021-04-21 07:10:05 UTC

R topics documented:

LearningStats-package *Elemental Descriptive and Inferential Statistics (LearningStats)*

Description

This package provides tools to teach students elemental Statistics. The main topics covered are Descriptive Statistics, Probability models (discrete and continuous variables) and Statistical Inference (confidence intervals and hypothesis tests).

Details

Main sections of LearningStats-package are:

A.- Data
B.- Descriptive Statistics
C.- Probability models
D.- Statistical Inference
E.- Regression

A.- Data

This section includes a function to read different file extensions and a dataset on health-related behaviours with 18 variables. The main advantage of this tool is that with just one single function most of the common file extensions can be imported into R.

`read.data`	Data Input
`sicri2018`	SICRI: information system on risk-taking behaviour

B.- Descriptive Statistics

The functions included in this section perform Descriptive Statistics by quantitatively describing or summarizing different characteristics from a sample. Graphical tools are also available.

`freq.pol`	Plot a Cumulative Frequency Polygon
`freq.table`	Frequency Table
`Histogram`	Plot a Histogram

C.- Probability models

In this section probability models for discrete and continuous variables are provided.

C.1-Discrete variables:

The user is allowed to display, with several options, the probability mass and/or distribution function for the following discrete distributions: Binomial, Discrete Uniform, Hypergeometric, Negative Binomial and Poisson.

`plotBinom`	Probability Mass and/or Distribution Function Representations associated with a Binomial Distribution
`plotDUnif`	Probability Mass and/or Distribution Function Representations associated with a Discrete Uniform Distribution
`plotHyper`	Probability Mass and/or Distribution Function Representations associated with a Hypergeometric Distribution
`plotNegBinom`	Probability Mass and/or Distribution Function Representations associated with a Negative Binomial Distribution
`plotPois`	Probability Mass and/or Distribution Function Representations associated with a Poisson Distribution

C.2-Continuous variables:

The user is allowed to display, with several options, the density, distribution and/or quantile functions for the following continuous distributions: Beta, Chi-squared, Exponential, F-Snedecor, Gamma, Normal, T-Student and Uniform.

`plotBeta`	Density Function, Distribution Function and/or Quantile Function Representations associated with a Beta Distribution
`plotChi`	Density Function, Distribution Function and/or Quantile Function Representations associated with a Chi-squared Distribution
`plotExp`	Density Function, Distribution Function and/or Quantile Function Representations associated with a Exponential Distribution
`plotFS`	Density Function, Distribution Function and/or Quantile Function Representations associated with a F-Snedecor Distribution
`plotGamma`	Density Function, Distribution Function and/or Quantile Function Representations associated with a Gamma Distribution
`plotNorm`	Density Function, Distribution Function and/or Quantile Function Representations associated with a Normal Distribution
`plotTS`	Density Function, Distribution Function and/or Quantile Function Representations associated with a T-Student Distribution
`plotUnif`	Density Function, Distribution Function and/or Quantile Function Representations associated with a Uniform Distribution

C.3-Illustrations:

Also in this section three common approximations between different distributions are illustrated.

The approximations considered are: the Normal approximation to Binomial, the Normal approximation to Poisson and the Poisson approximation to Binomial.

AproxBinomNorm	Illustration of the Normal Approximation to Binomial
AproxPoisNorm	Illustration of the Normal Approximation to Poisson
AproxBinomPois	Illustration of the Poisson Approximation to Binomial

D.- Statistical Inference

This section includes functions to perform Statistical Inference (confidence intervals and hypothesis testing) with one or two populations and also for categorical data.

D.1-Confidence intervals:

The functions included here provide pointwise and confidence interval estimation for different population parameters. One or two populations are supported.

One population:

Mean.CI	Confidence Interval for the Mean of a Normal Population
proportion.CI	Large Sample Confidence Interval for a Population Proportion
variance.CI	Confidence Interval for the Variance and the Standard Deviation of a Normal Population

Two populations:

diffmean.CI	Confidence Interval for the Difference between the Means of Two Normal Populations
diffproportion.CI	Large Sample Confidence Interval for the Difference between Two Population Proportions
diffvariance.CI	Confidence Interval for the Ratio between the Variances of Two Normal Populations

D.2-Hypothesis tests:

This sections allows to compute hypothesis tests for different population parameters (mean, variance and proportion) in one or two populations. The scenarios covered here are those mentioned in the Confidence Interval section as well as a Chi-squared independence test.

One population:

Mean.test	One Sample Mean Test of a Normal Population
proportion.test	Large Sample Test for a Population Proportion
variance.test	One Sample Variance Test of a Normal Population

Two populations:

diffmean.test	Two Sample Mean Test of Normal Populations
diffproportion.test	Two Sample Proportion Test
diffvariance.test	Two Sample Variance Test of Normal Populations

Categorical data:

indepchisq.test	Chi-squared Independence Test for Categorical Data

E.- Regression

This section includes a function to describe the relationship between two continuous variables through a simple linear regression model, providing the R-squared coefficient.

plotReg	Representation of a Linear Regression Model

AproxBinomNorm *Illustration of the Normal approximation to Binomial*

Description

When certain conditions are met (see Details), the Binomial distribution can be approximated by the Normal one. The function `AproxBinomNorm` illustrates this fact by plotting the mass diagram corresponding with the discrete distribution (parameters are given by the user) on which the associated Normal density function is also displayed.

Usage

```
AproxBinomNorm(n, p, legend = TRUE, xlab = "", ylab = "Probability",
  main = "Normal approximation to Binomial", col.fill = "grey",
  col.line = "red", lwd = 2)
```

Arguments

n	number of independent Bernouilli trials.
p	probability of success associated with the Bernouilli trial.
legend	logical argument indicating whether to display the legend on the plot or not, default to TRUE.
xlab	x-axis label; default to empty.
ylab	y-axis label; default to "Probability."
main	title; default to "Normal approximation to Binomial".

`col.fill`	colour to fill-in the bars; default to grey.
`col.line`	colour to draw the line of the Normal density; default to red.
`lwd`	line width for the Normal density, a positive number; default to 2.

Details

The approximation is accurate only if one of these three conditions is met:

- p in (0.1, 0.9) and n>=30,

- p in [0,0.1] and np>5,

- p in [0.9,1] and n(1-p)>5.

Value

This function is called for the side effect of drawing the plot.

Examples

```
n=45; p=0.4
AproxBinomNorm(n,p)
AproxBinomNorm(n,p,col.fill="blue",col.line="orange")
AproxBinomNorm(n,p,legend=FALSE)
```

`AproxBinomPois` *Illustration of the Poisson approximation to Binomial*

Description

`AproxBinomPois` represents the probability mass associated with a Binomial distribution with certain parameters `n` and `p` joint with the Poisson distribution with mean equal to `np`. Note that the Binomial distribution can be approximated by a Poisson distribution when certain conditions are met (see Details).

Usage

```
AproxBinomPois(n, p, xlab = "x", ylab = "Probability Mass",
  main = "Poisson approximation to Binomial distribution", col1 = "grey",
  col2 = "red")
```

Arguments

`n`	number of independent Bernoulli trials.
`p`	probability of success associated with the Bernoulli trial.
`xlab`	x-axis label; default to "x".
`ylab`	y-axis label; default to "Probability Mass".

`main`	an overall title for the plot; default to "Poisson approximation to Binomial distribution".
`col1`	a single colour associated with the Binomial probability mass function; default to "grey".
`col2`	a single colour associated with the Poisson probability mass function; default to "red".

Details

The approximation is accurate only if one of these conditions is met:

- `p` in (0,0.1), `n>=30` and `np<5`,

- `p` in (0.9,1), `n>=30` and `n(1-p)<5`. Note that given X1 a Binomial distribution with parameters `n` and `p`, and X2 a Binomial distribution with parameters `n` and `1-p`, it follows that P(X1=a)=P(X2=n-a). Then, the variable X2 can be approximated to a Poisson distribution with parameter `lambda=n(1-p)` and this Poisson distribution can be used in order to approximate the mass probability function associated with X1.

Value

This function is called for the side effect of drawing the plot.

Examples

```
n=50;p=0.93
AproxBinomPois(n,p)
n=100;p=0.03
AproxBinomPois(n,p)
```

`AproxPoisNorm` *Illustration of the Normal approximation to Poisson*

Description

When certain conditions are met (see Details), the Poisson distribution can be approximated by the Normal one. The function `AproxPoisNorm` illustrates this fact by plotting the mass diagram corresponding with the discrete distribution (parameter is given by the user) on which the associated Normal density function is also displayed.

Usage

```
AproxPoisNorm(lambda, legend = TRUE, xlab = "", ylab = "Probability",
  main = "Normal approximation to Poisson", col.fill = "grey",
  col.line = "red", lwd = 2)
```

Arguments

`lambda`	mean of the Poisson distribution.
`legend`	logical argument indicating whether to display the legend on the plot or not, default to TRUE.
`xlab`	x-axis label; default to empty.
`ylab`	y-axis label; default to "Probability".
`main`	title; default to "Normal approximation to Poisson".
`col.fill`	colour to fill the bars; default to grey.
`col.line`	colour to draw the line of the Normal density; default to red.
`lwd`	line width for the Normal density, a positive number; default to 2.

Details

The approximation is accurate only if lambda>=10.

Value

This function is called for the side effect of drawing the plot.

Examples

```
lambda=15
AproxPoisNorm(lambda)
AproxPoisNorm(lambda,col.fill="blue",col.line="orange")
AproxPoisNorm(lambda,legend=FALSE)
```

`BoxPlot` *Boxplot Representation*

Description

The function `BoxPlot` displays a boxplot representation of a given sample.

Usage

```
BoxPlot(x, col = "white", main = "Boxplot representacion", ylab = "",
  legend = TRUE)
```

Arguments

`x`	a numeric vector containing the sample to plot the boxplot representation.
`col`	a single colour to fill the boxplot representation; default to "white".
`main`	a main title for the boxplot; default to "Boxplot representation".
`ylab`	y-axis label; default to empty.
`legend`	logical value; if TRUE (default), details about boxplot representation are given.

Details

The quantiles needed to obtain this representation are computed using the function `sample.quantile`.

Value

This function is called for the side effect of drawing the plot.

Examples

```
x=c(5,-5,rnorm(40))
BoxPlot(x,col="pink")
```

`diffmean.CI` *Confidence Interval for the Difference between the Means of Two Normal Populations.*

Description

`diffmean.CI` provides a pointwise estimation and a confidence interval for the difference between the means of two Normal populations in different scenarios: population variances known or unknown, population variances assumed equal or not, and paired or independent populations.

Usage

```
diffmean.CI(x1, x2, sigma1 = NULL, sigma2 = NULL, sc1 = NULL,
  sc2 = NULL, s1 = NULL, s2 = NULL, n1 = NULL, n2 = NULL,
  paired = FALSE, var.equal = FALSE, conf.level)
```

Arguments

`x1`	numeric vector or value corresponding with either one of the samples or the sample mean.
`x2`	numeric vector or value corresponding with either one of the samples or the sample mean.
`sigma1`	if known, a single numeric value corresponding with one of the population standard deviation.
`sigma2`	if known, a single numeric value corresponding with the other population standard deviation.
`sc1`	a single numeric value corresponding with the cuasi-standard deviation of one sample.
`sc2`	a single numeric value corresponding with the cuasi-standard deviation of the other sample.
`s1`	a single numeric value corresponding with the standard deviation of one sample.
`s2`	a single numeric value corresponding with the standard deviation of the other sample.

n1	a single positive integer value corresponding with the size of one sample; not needed if the sample is provided.
n2	a single positive integer value corresponding with the size of the other sample; not needed if the sample is provided.
paired	logical value indicating whether the populations are paired or independent; default to FALSE.
var.equal	logical value indicating whether to treat the two variances as being equal. If TRUE then the pooled variance is used to estimate the variance, otherwise the Dixon and Massey approximation to the degrees of freedom is used; default to FALSE.
conf.level	a single numeric value corresponding with the confidence level of the interval; must be a value in (0,1).

Details

If sigma1 and sigma2 are given, known population variances formula is applied; the unknown one is used in other case.

If paired is TRUE then both x1 and x2 must be specified and their sample sizes must be the same. If paired is null, then it is assumed to be FALSE.

For var.equal=TRUE, the formula of the pooled variance is $\frac{(n1-1)sc1^2+(n2-1)sc2^2}{n1+n2-2}$.

Value

A list containing the following components:

estimate	numeric value corresponding with the difference between the sample means.
CI	a numeric vector of length two containing the lower and upper bounds of the confidence interval.

Independently on the user saving those values, the function provides a summary of the result on the console.

Examples

```
#Given unpaired samples with known population variance
dat1=rnorm(20,mean=2,sd=1);dat2=rnorm(30,mean=2,sd=1.5)
diffmean.CI(dat1,dat2,sigma1=1,sigma2=1.5,conf.level=0.9)

#Given unpaired samples with unknown but equal population variances
dat1=rnorm(20,mean=2,sd=1);dat2=rnorm(30,mean=2,sd=1)
diffmean.CI(dat1,dat2,paired=FALSE,var.equal=TRUE,conf.level=0.9)

#Given the characteristics of unpaired samples with unknown and different population variances
dat1=rnorm(20,mean=2,sd=1);dat2=rnorm(30,mean=2,sd=1)
x1=mean(dat1);x2=mean(dat2);sc1=sd(dat1);sc2=sd(dat2);n1=length(dat1);n2=length(dat2)
diffmean.CI(x1,x2,sc1=sc1,sc2=sc2,n1=n1,n2=n2,paired=FALSE,var.equal=FALSE,conf.level=0.9)

#Given paired samples
dat1=rnorm(20,mean=2,sd=1);dat2=dat1+rnorm(20,mean=0,sd=0.5)
```

```
diffmean.CI(dat1,dat2,paired=TRUE,conf.level=0.9)
```

`diffmean.test` *Two Sample Mean Test of Normal Populations*

Description

`diffmean.test` allows to compute hypothesis tests about two population means. The difference between the means of two Normal populations is tested in different scenarios: known or unknown variance, variances assumed equal or different and paired or independent populations.

Usage

```
diffmean.test(x1, x2, sigma1 = NULL, sigma2 = NULL, sc1 = NULL,
  sc2 = NULL, s1 = NULL, s2 = NULL, n1 = NULL, n2 = NULL,
  var.equal = FALSE, paired = FALSE, alternative = "two.sided",
  alpha = 0.05, plot = TRUE, lwd = 1)
```

Arguments

`x1`	a numeric vector of data values or, if single number, estimated mean.
`x2`	a numeric vector of data values or, if single number, estimated mean.
`sigma1`	if known, a single numeric value corresponding with one of the population standard deviation.
`sigma2`	if known, a single numeric value corresponding with the other population standard deviation.
`sc1`	cuasi-standard deviation of sample `x1`. By default computes the cuasi-standard deviation of argument `x1`.
`sc2`	cuasi-standard deviation of sample `x2`. By default computes the cuasi-standard deviation of argument `x2`.
`s1`	sample standard deviation of sample x1. Defaults to NULL, if provided, it computes the cuasi-standard deviation.
`s2`	sample standard deviation of sample x2. Defaults to NULL, if provided, it computes the cuasi-standard deviation.
`n1`	sample size of `x1`. By default length of argument `x1`.
`n2`	sample size of `x2`. By default length of argument `x2`.
`var.equal`	a logical indicating whether to treat the two variances as being equal. Defaults to FALSE.
`paired`	a logical indicating whether the samples are paired. Defaults to FALSE, if TRUE, then both x1 and x2 must be the same length.
`alternative`	a character string specifying the alternative hypothesis, must be one of "`two.sided`" (default), "`greater`" or "`less`".

`alpha`	single number in (0,1), corresponding with the significance level.
`plot`	a logical value indicating whether to display a graph including the test statistic value for the sample, its distribution, the rejection region and p-value.
`lwd`	single number indicating the line width of the plot.

Details

If sigma1 and sigma2 are given, known population variances formula is applied; the unknown one is used in other case.

If paired is TRUE then both x1 and x2 must be specified and their sample sizes must be the same. If paired is null, then it is assumed to be FALSE.

For var.equal=TRUE, the formula of the pooled variance is $\frac{(n1-1)sc1^2+(n2-1)sc2^2}{n1+n2-2}$.

Value

A list with class "`lstest`" and "`htest`" containing the following components:

`statistic`	the value of the test statistic.
`parameter`	the degrees of freedom of the statistic's distribution. NULL for the Normal distribution.
`p.value`	the p-value of the test.
`estimate`	the estimated difference in means.
`null.value`	the value specified by the null.
`alternative`	a character string describing the alternative.
`method`	a character string indicating the method used.
`data.name`	a character string giving the names of the data.
`alpha`	the significance level.
`dist.name`	a character string indicating the distribution of the test statistic.
`statformula`	a character string with the statistic's formula.
`reject.region`	a character string with the reject region.

Examples

```
x1 <- rnorm(40, mean = 1.5, sd = 2)
x2 <- rnorm(60, mean = 2, sd = 2)
#equal variances
diffmean.test(x1, x2, var.equal = TRUE)
diffmean.test(mean(x1), mean(x2),
              sc1 = sd(x1), sc2 = sd(x2),
              n1 = 40, n2 = 60, var.equal = TRUE)
x3 <- rnorm(60, mean = 2, sd = 1.5)
#different variances
diffmean.test(x1, x3)
#known standard deviation
diffmean.test(x1, x3, sigma1 = 2, sigma2 = 1.5)
x4 <- x1 + rnorm(40, mean = 0, sd = 0.1)
#paired samples
diffmean.test(x1, x4, paired = TRUE)
```

`diffproportion.CI` *Large Sample Confidence Interval for the Difference between Two Population Proportions*

Description

`diffproportion.CI` provides a pointwise estimation and a confidence interval for the difference between two population proportions.

Usage

```
diffproportion.CI(x1, x2, n1, n2, conf.level)
```

Arguments

`x1`	a single numeric value corresponding with either the proportion estimate or the number of successes of one of the samples.
`x2`	a single numeric value corresponding with either the proportion estimate or the number of successes of the other sample.
`n1`	a single positive integer value corresponding with one sample size.
`n2`	a single positive integer value corresponding with the other sample size.
`conf.level`	a single numeric value corresponding with the confidence level of the interval; must be a value in (0,1).

Details

Counts of successes and failures must be nonnegative and hence not greater than the corresponding numbers of trials which must be positive. All finite counts should be integers. If the number of successes are given, then the proportion estimate is computed.

Value

A list containing the following components:

`estimate`	a numeric value corresponding with the difference between the two sample proportions.
`CI`	a numeric vector of length two containing the lower and upper bounds of the confidence interval.

Independently on the user saving those values, the function provides a summary of the result on the console.

Examples

```
#Given the sample proportion estimate
diffproportion.CI(0.3,0.4,100,120,conf.level=0.95)

#Given the number of successes
diffproportion.CI(30,48,100,120,conf.level=0.95)

#Given in one sample the number of successes and in the other the proportion estimate
diffproportion.CI(0.3,48,100,120,conf.level=0.95)
```

---

diffproportion.test *Two Sample Proportion Test*

Description

`diffproportion.test` allows to compute hypothesis tests about two population proportions.

Usage

```
diffproportion.test(x1, x2, n1, n2, alternative = "two.sided",
  alpha = 0.05, plot = TRUE, lwd = 1)
```

Arguments

`x1`	a single numeric value corresponding with either the proportion estimate or the number of successes of one of the samples.
`x2`	a single numeric value corresponding with either the proportion estimate or the number of successes of the other sample.
`n1`	a single positive integer value corresponding with one sample size.
`n2`	a single positive integer value corresponding with the other sample size.
`alternative`	a character string specifying the alternative hypothesis, must be one of "`two.sided`" (default), "`greater`" or "`less`".
`alpha`	single number in (0,1) corresponding with the significance level.
`plot`	a logical value indicating whether to display a graph including the test statistic value for the sample, its distribution, the rejection region and p-value.
`lwd`	single number indicating the line width of the plot.

Details

Counts of successes and failures must be nonnegative and hence not greater than the corresponding numbers of trials which must be positive. All finite counts should be integers. If the number of successes is given, then the proportion estimate is computed.

Value

A list with class "lstest" and "htest" containing the following components:

statistic	the value of the test statistic.
parameter	the sample size n1.
p.value	the p-value of the test.
estimate	the difference of sample proportions.
null.value	the value specified by the null.
alternative	a character string describing the alternative.
method	a character string indicating the method used.
data.name	a character string giving the names of the data.
alpha	the significance level.
dist.name	a character string indicating the distribution of the test statistic.
statformula	a character string with the statistic's formula.
reject.region	a character string with the reject region.

Examples

```
x1 <- rbinom(1, 120, 0.6)
x2 <- rbinom(1, 100, 0.6)
diffproportion.test(x1 = x1, x2 = x2, n1 = 120, n2 = 100)
diffproportion.test(x1 = 0.6, x2 = 0.65, n1 = 120, n2 = 100)
```

---

diffvariance.CI   *Confidence Interval for the Ratio Between the Variances of Two Normal Populations*

Description

diffvariance.CI provides a pointwise estimation and a confidence interval for the ratio of Normal population variances in both scenarios: known and unknown population mean.

Usage

```
diffvariance.CI(x1 = NULL, x2 = NULL, s1 = NULL, s2 = NULL,
  sc1 = NULL, sc2 = NULL, smu1 = NULL, smu2 = NULL, mu1 = NULL,
  mu2 = NULL, n1 = NULL, n2 = NULL, conf.level)
```

Arguments

`x1`	a numeric vector containing the sample of one population.
`x2`	a numeric vector containing the sample of the other population.
`s1`	a single numeric value corresponding with the sample standard deviation of the first sample.
`s2`	a single numeric value corresponding with the sample standard deviation of the second sample.
`sc1`	a single numeric value corresponding with the cuasi-standard deviation of the first sample.
`sc2`	a single numeric value corresponding with the cuasi-standard deviation of the second sample.
`smu1`	if known, a single numeric value corresponding with the estimation of the standard deviation of the first sample.
`smu2`	if known, a single numeric value corresponding with the estimation of the standard deviation of the second sample.
`mu1`	if known, a single numeric corresponding with the mean of one population.
`mu2`	if known, a single numeric value corresponding with the mean of the other population.
`n1`	a single positive integer corresponding with the size of one sample.
`n2`	a single positive integer corresponding with the size of the other sample.
`conf.level`	is the confidence level of the interval; must be a value in (0,1).

Details

The formula interface is applicable when the user provides the sample and when the user provides the value of the sample characteristics (sample mean, cuasi-standard deviation or sample standard deviation, and sample size). Moreover, when mu1, smu1, mu2 or smu2 are provided, the function performs the procedure with known population means, and unknown in other case.

Value

A list containing the following components:

`var.estimate`	numeric value corresponding with the ratio of cuasi-variances for unknown population mean, and the ratio of the sample variances for known population mean.
`sd.estimate`	numeric value corresponding with the ratio of cuasi-standard deviations for unknown population mean and the ratio of the sample standard deviations for known population mean.
`CI.var`	a numeric vector of length two containing the lower and upper bounds of the confidence interval for the population variance.
`CI.sd`	a numeric vector of length two containing the lower and upper bounds of the confidence interval for the population standard deviation.

Independently on the user saving those values, the function provides a summary of the result on the console.

Examples

```
#Given the samples with known population means
dat1=rnorm(20,mean=2,sd=1); dat2=rnorm(30,mean=3,sd=1)
diffvariance.CI(x1=dat1,x2=dat2,mu1=2,mu2=3,conf.level=0.95)

#Given the sample standard deviations with known population means
dat1=rnorm(20,mean=2,sd=1); dat2=rnorm(30,mean=3,sd=1)
smu1=Smu(dat1,mu=2);smu2=Smu(dat2,mu=3)
diffvariance.CI(smu1=smu1,smu2=smu2,n1=20,n2=30,conf.level=0.95)

#Given the samples with unknown population means
dat1=rnorm(20,mean=2,sd=1); dat2=rnorm(30,mean=3,sd=1)
diffvariance.CI(x1=dat1,x2=dat2,conf.level=0.95)

#Given the sample standard deviations with unknown population means
dat1=rnorm(20,mean=2,sd=1); dat2=rnorm(30,mean=3,sd=1)
diffvariance.CI(s1=(19/20)*sd(dat1),s2=(29/30)*sd(dat2),n1=20,n2=30,conf.level=0.95)
```

---

diffvariance.test *Two Sample Variance Test of Normal Populations*

Description

`diffvariance.test` allows to compute hypothesis tests about two population variances in both scenarios: known and unknown population mean.

Usage

```
diffvariance.test(x1 = NULL, x2 = NULL, s1 = NULL, s2 = NULL,
  sc1 = NULL, sc2 = NULL, smu1 = NULL, smu2 = NULL, mu1 = NULL,
  mu2 = NULL, n1 = NULL, n2 = NULL, alternative = "two.sided",
  alpha = 0.05, plot = TRUE, lwd = 1)
```

Arguments

`x1`	a numeric vector containing the sample of one population.
`x2`	a numeric vector containing the sample of the other population.
`s1`	a single numeric value corresponding with the sample standard deviation of the first sample.
`s2`	a single numeric value corresponding with the sample standard deviation of the second sample.
`sc1`	a single numeric value corresponding with the cuasi-standard deviation of the first sample.
`sc2`	a single numeric value corresponding with the cuasi-standard deviation of the second sample.

smu1	if known, a single numeric value corresponding with the estimation of the standard deviation of the first sample.
smu2	if known, a single numeric value corresponding with the estimation of the standard deviation of the second sample.
mu1	if known, a single numeric corresponding with the mean of one population.
mu2	if known, a single numeric value corresponding with the mean of the other population.
n1	a single number indicating the sample size of x1. By default length of argument x1.
n2	a single number indicating the sample size of x2. By default length of argument x2.
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less".
alpha	single number between 0 and 1, significance level.
plot	a logical value indicating whether to display a graph including the test statistic value for the sample, its distribution, the rejection region and p-value.
lwd	single number indicating the line width of the plot.

Details

The formula interface is applicable when the user provides the sample(s) or values of the sample characteristics (cuasi-standard deviation or sample standard deviation). When mu1 and mu2 or smu1 and smu2 are provided, the function performs the procedure with known population means.

Value

A list with class "lstest" and "htest" containing the following components:

statistic	the value of the test statistic.
parameter	the degrees of the freedom of the F distribution of the test statistic.
p.value	the p-value of the test.
estimate	the ratio of the cuasi-variances of x1 and x2.
null.value	the value specified by the null.
alternative	a character string describing the alternative.
method	a character string indicating the method used.
data.name	a character string giving the names of the data.
alpha	the significance level.
dist.name	a character string indicating the distribution of the test statistic.
statformula	a character string with the statistic's formula.
reject.region	a character string with the reject region.

Examples

```
x1 <- rnorm(40, mean = 1, sd = 2)
x2 <- rnorm(60, mean = 2, sd = 1.5)
# unknown population mean
diffvariance.test(x1, x2)
diffvariance.test(x1, sc2 = sd(x2), n2 = length(x2))
diffvariance.test(sc1 = sd(x1), sc2 = sd(x2), n1 = length(x1), n2 = length(x2))
# known population mean
diffvariance.test(x1, x2, mu1 = 1, mu2 = 2)
smu1 <- Smu(x1, mu = 1); smu2 <- Smu(x2, mu = 2)
diffvariance.test(smu1 = smu1, smu2 = smu2, n1 = length(x1), n2 = length(x2))
```

---

freq.pol *Plot a Cumulative Frequency Polygon*

Description

The function `freq.pol` computes a cumulative frequency polygon of a given sample.

Usage

```
freq.pol(x, freq = FALSE, col = "black", lwd = 2, main = "autom",
  xlab = "", ylab = "autoy", bar = TRUE, fill = TRUE,
  col.fill = "pink")
```

Arguments

`x`	a numeric vector containing the sample to compute the cumulative polygon.
`freq`	logical value; if TRUE, the cumulative polygon uses absolute frequencies; if FALSE, relative frequencies are used.
`col`	colour to be used in the polygon line; default to black.
`lwd`	a single numeric value corresponding with the width to be used in the polygon line; default to 2
`main`	main title; by default to "Polygon of cumulative absolute frequencies" or "Polygon of cumulative relative frequencies" depending on the value of the argument freq, TRUE or FALSE respectively.
`xlab`	x-axis label; by default to empty .
`ylab`	y-axis label; by default to "Cumulative absolute frequencies" or "Cumulative relative frequencies" depending on the value of the argument freq, TRUE or FALSE respectively.
`bar`	logical value; if TRUE (default), bars are plotted underneath the polygon line.
`fill`	logical value; if TRUE bars are filled with colour set in col.fill (if not given pink is chosen); FALSE (default) unless col.fill is given.
`col.fill`	colour to be used to fill the bars; if not given and fill=TRUE, set to pink.

Details

The sample must be numeric and coming from a continuous variable.

The procedure used to define the intervals for the frequency table and the bars (if plotted) is the same as used for the histogram performed in this package (see ?Histogram).

Value

A list containing the following components:

`ni`	a numeric vector containing the absolute frequencies.
`fi`	a numeric vector containing the relative frequencies.
`Ni`	a numeric vector containing the absolute cumulative frequencies.
`Fi`	a numeric vector containing the relative cumulative frequencies.
`tab`	the frequency table.

Independently on the user saving those values, the function provides the frequency table on the console.

Examples

```
x=rnorm(10)
freq.pol(x)

freq.pol(x,freq=TRUE,fill=TRUE,col.fill="yellow")
```

`freq.table` *Frequency Table*

Description

The function `freq.table` computes a frequency table with absolute and relative frequencies (for non-ordered variables); and with those as well as their cumulative counterparts (for ordered variables).

Usage

```
freq.table(x, cont, ord = NULL)
```

Arguments

`x`	a vector containing the sample provided to compute the frequency table
`cont`	logical; if TRUE, the sample comes from a continuous variable; if FALSE the sample is treated as coming from a discrete or categorical variable.
`ord`	if needed, character vector containing the ordered categories' names of the ordinal variable.

Details

The procedure used to define the intervals for the frequency table in the continuous case is the same as used for the histogram (see ?Histogram).

Value

A list containing the following components:

`ni`	a numeric vector containing the absolute frequencies.
`fi`	a numeric vector containing the relative frequencies.
`Ni`	a numeric vector containing the absolute cumulative frequencies.
`Fi`	a numeric vector containing the relative cumulative frequencies.
`di`	if cont=TRUE, a vector containing the frequency density.
`tab`	the frequency table.

The values of the cumulative frequencies (Ni and Fi) are only computed and provided when the variable of interest is ordered. If the user does not save those values, the function provides the list on the console.

Examples

```
#Nominal variable
x=sample(c("yellow","red","blue","green"),size=20,replace=TRUE)
freq.table(x,cont=FALSE)

#Ordinal variable
x=sample(c("high","small","medium"),size=20,replace=TRUE)
freq.table(x,cont=FALSE,ord=c("small","medium","high"))

#Discrete variable
x=sample(1:5,size=20,replace=TRUE)
freq.table(x,cont=FALSE)

#Continuous variable
x=rnorm(20)
freq.table(x,cont=TRUE)
```

`Histogram` *Plot a Histogram*

Description

The function `Histogram` plots a histogram of a given sample.

Usage

```
Histogram(x, freq = FALSE, col.fill = "grey", main = "autom",
  xlab = "", ylab = "autoy")
```

Arguments

`x`	a numeric vector containing the sample provided to compute the histogram.
`freq`	a single logical value; if TRUE, the histogram graphic uses absolute frequencies; if FALSE (default), a histogram of area 1 (density) is plotted.
`col.fill`	a single colour to be used to fill the bars; default to grey.
`main`	main title, by default "Histogram" or "Histogram of area 1" depending on the value of the argument freq, TRUE or FALSE respectively.
`xlab`	x-axis label; by default empty.
`ylab`	y-axis label; by default "Frequency" or "Density" depending on the value of the argument freq, TRUE or FALSE respectively.

Details

The procedure to construct the histogram is detailed below:

- number of intervals: the closest integer to sqrt(n);

- amplitude of each interval: the range of the sample divided by the number of intervals, i.e., the breaks are equidistant and rounded to two decimals;

- height of each bar: by default (freq=FALSE) the plotted histogram is a density (area 1); if freq=TRUE, then the values of the bars are the absolute frequencies.

Value

A list containing the following components:

`ni`	a numeric vector containing the absolute frequencies.
`fi`	a numeric vector containing the relative frequencies.
`Ni`	a numeric vector containing the absolute cumulative frequencies.
`Fi`	a numeric vector containing the relative cumulative frequencies.
`tab`	the frequency table.

Independently on the user saving those values, the function provides the frequency table on the console.

Examples

```
x=rnorm(10)
Histogram(x)
Histogram(x,freq=TRUE)
Histogram(x,freq=TRUE,col="pink")
```

`indepchisq.test` *Chi-squared Independence Test for Categorical Data.*

Description

`indepchisq.test` allows to computes Chi-squared independence hypothesis test for two categorical values.

Usage

```
indepchisq.test(Oij, x, y, alpha = 0.05, plot = TRUE, lwd = 1)
```

Arguments

`Oij`	observed frequencies. A numeric matrix, a table or a data.frame with the observed frequencies can be passed. If missing, arguments `x` and `y` must be supplied.
`x`	a vector (numeric or character) or factor with the first categorical variable.
`y`	a vector (numeric or character) or factor with the second categorical variable. It should be of the same length as `x`.
`alpha`	a single number in (0,1), significance level.
`plot`	a logical indicating whether to plot the rejection region and p-value.
`lwd`	a single number indicating the line width of the plot.

Details

The expected frequencies are calculated as follows

$$E_{ij} = \frac{n_{i\bullet} \times n_{\bullet j}}{n},$$

and the test statistic is given by

$$T = \sum_{i,j} \frac{(n_{ij} - E_{ij})^2}{E_{ij}},$$

$T \in \chi^2_{(r-1)(s-1)}$, where n is the number of observations, $n_{i\bullet}$ is the marginal frequency of category i of variable x, $n_{\bullet j}$ is the marginal frequency of category j of variable y, r is the number of categories in variable x and s the number of categories in variable y.

The null hypothesis is rejected when $T > \chi^2_{(r-1)(s-1),1-\alpha}$, where $\chi^2_{(r-1)(s-1),1-\alpha}$ is the $1-\alpha$ quantile of a χ^2 distribution with $(r-1)(s-1)$ degrees of freedom.

Value

A list with class "lstest" and "htest" containing the following components:

statistic	the value of the test statistic.
parameter	the degrees of freedom of the statistic's distribution.
p.value	the p-value of the test.
estimate	a numeric matrix with the estimated frequencies Eij.
method	a character string indicating the method used.
data.name	a character string giving the names of the data.
alpha	the significance level.
dist.name	a character string indicating the distribution of the test statistic.
statformula	a character string with the statistic's formula.
reject.region	a character string with the reject region.
obs.freq	a numeric matrix with the observed frequencies Oij.

Examples

```
Oij <- matrix(c( 20,    8,
               934, 1070,
               113,   92), ncol = 2, byrow = TRUE)
indepchisq.test(Oij)
```

Mean.CI — *Confidence Interval for the Mean of a Normal Population*

Description

Mean.CI provides a pointwise estimation and a confidence interval for the mean of a Normal population in both scenarios: known and unknown population variance.

Usage

```
Mean.CI(x, sigma = NULL, sc = NULL, s = NULL, n = NULL, conf.level)
```

Arguments

x	numeric value or vector containing either the sample or the sample mean.
sigma	if known, a single numeric value corresponding with the population standard deviation.
sc	a single numeric value corresponding with the cuasi-standard deviation; not needed if if the sample is provided.
s	a single numeric value corresponding with the sample standard deviation; not needed if the sample is provided.
n	a single positive integer corresponding with the sample size; not needed if the sample is provided.
conf.level	a single value corresponding with the confidence level of the interval; must be a value in (0,1).

Details

The formula interface is applicable when the user provides the sample and also when the user provides the value of the sample characteristics (sample mean, cuasi-standard deviation or sample standard deviation, jointly with the sample size).

Value

A list containing the following components:

`estimate` a numeric value corresponding with the sample mean.

`CI` a numeric vector of length two containing the lower and upper bounds of the confidence interval.

Independently on the user saving those values, the function provides a summary of the result on the console.

Examples

```
#Given the sample with known population variance
dat=rnorm(20,mean=2,sd=1)
Mean.CI(dat, sigma=1, conf.level=0.95)

#Given the sample with unknown population variance
dat=rnorm(20,mean=2,sd=1)
Mean.CI(dat, conf.level=0.95)

#Given the sample mean with known population variance:
dat=rnorm(20,mean=2,sd=1)
Mean.CI(mean(dat),sigma=1,n=20,conf.level=0.95)

#Given the sample mean with unknown population variance:
dat=rnorm(20,mean=2,sd=1)
Mean.CI(mean(dat),sc=sd(dat),n=20,conf.level=0.95)
```

`Mean.test` *One Sample Mean Test of a Normal Population*

Description

`Mean.test` allows to compute hypothesis tests for a Normal population mean in both scenarios: known and unknown population variance.

Usage

```
Mean.test(x, mu0, sigma = NULL, sc = NULL, s = NULL, n = NULL,
  alternative = "two.sided", alpha = 0.05, plot = TRUE, lwd = 1)
```

Arguments

`x`	a numeric vector of data values or, if single number, estimated mean.
`mu0`	a single number corresponding with the mean to test.
`sigma`	population standard deviation. Defaults to NULL, if specified, a t-test with known variance will be performed.
`sc`	cuasi-standard deviation of sample x. By default computes the cuasi-standard deviation of argument x.
`s`	sample standard deviation of sample x. Defaults to NULL, if provided, it computes the cuasi-standard deviation.
`n`	sample size. By default length of argument x.
`alternative`	a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"` or `"less"`.
`alpha`	a single number in (0,1), significance level.
`plot`	a logical value indicating whether to display a graph including the test statistic value for the sample, its distribution, the rejection region and p-value.
`lwd`	single number indicating the line width of the plot.

Details

The formula interface is applicable when the user provides the sample and also when the user provides the value of the sample characteristics (sample mean, cuasi-standard deviation or sample standard deviation, jointly with the sample size).

Value

A list with class `"lstest"` and `"htest"` containing the following components:

`statistic`	the value of the test statistic.
`parameter`	the degrees of freedom of the statistic's distribution. NULL for the Normal distribution.
`p.value`	the p-value of the test.
`estimate`	the sample mean.
`null.value`	the value specified by the null.
`alternative`	a character string describing the alternative.
`method`	a character string indicating the method used.
`data.name`	a character string giving the names of the data.
`alpha`	the significance level.
`dist.name`	a character string indicating the distribution of the test statistic.
`statformula`	a character string with the statistic's formula.
`reject.region`	a character string with the reject region.
`unit`	a character string with the units.

Examples

```
x <- rnorm(50, mean = 4, sd = 2)
#unknown sigma
Mean.test(x, mu0 = 3.5)
Mean.test(mean(x), sc = sd(x),  n = length(x), mu0 = 3.5)
#known sigma
Mean.test(x, mu0 = 3.5, sigma = 2)
```

---

plotBeta — *Density Function, Distribution Function and/or Quantile Function Representations associated with a Beta Distribution*

Description

`plotBeta` represents density, distribution and/or quantile functions associated with a Beta distribution with parameters `shape1` and `shape2`.

Usage

```
plotBeta(shape1, shape2, type = "b", col = "black")
```

Arguments

`shape1, shape2`	parameters of the Beta distribution (mean equal to `shape1/(shape1+shape2)`
`type`	a character string giving the type of plot desired. The following values are possible: "b" (default) for density function, distribution function and quantile function representations together, "dis" for distribution function representation, "den" for density function representation and "q" for quantile function representation.
`col`	a single colour associated with the different representations; default to "black".

Value

This function is called for the side effect of drawing the plot.

Examples

```
shape1=1;shape2=1
plotBeta(shape1,shape2)
plotBeta(shape1,shape2,col="red")
plotBeta(shape1,shape2,type="q")
plotBeta(shape1,shape2,type="dis")
plotBeta(shape1,shape2,type="den")
```

`plotBinom` *Probability Mass and/or Distribution Function Representations associated with a Binomial Distribution*

Description

`plotBinom` represents the probability mass and/or the distribution function associated with a Binomial distribution with certain parameters `n` and `p`.

Usage

```
plotBinom(n, p, type = "b", col = "grey")
```

Arguments

`n`	the number of independent Bernoulli trials.
`p`	the probability of success associated with the Bernoulli trial.
`type`	a character string giving the type of desired plot. The following values are possible: "b" (default) for probability mass function and distribution function representations together, "d" for distribution function representation and "p" for probability mass function representation.
`col`	a single colour associated with the probability mass function representation; default to "grey".

Details

Note that if `n=1`, the Binomial distribution is also known as Bernoulli distribution.

Value

A matrix containing the probability mass and the distribution function associated with each point of the support of a Binomial distribution with parameters `n` and `p`.

This function is called for the side effect of drawing the plot.

Examples

```
n=10;p=0.3
plotBinom(n,p,type="d")
plotBinom(n,p,type="p",col="pink")
plotBinom(n,p)
```

plotChi	*Density Function, Distribution Function and/or Quantile Function Representations associated with a Chi-squared Distribution*

Description

`plotChi` represents density, distribution and/or quantile functions associated with a Chi-squared distribution with `df` degrees of freedom.

Usage

```
plotChi(df, type = "b", col = "black")
```

Arguments

`df`	the degrees of freedom of the Chi-squared distribution.
`type`	a character string giving the type of desired plot. The following values are possible: "b" (default) for density function, distribution function and quantile function representations together, "dis" for distribution function representation, "den" for density function representation and "q" for quantile function representation.
`col`	a single colour associated with the different representations; default to "black".

Value

This function is called for the side effect of drawing the plot.

Examples

```
df=10
plotChi(df)
plotChi(df,col="red")
plotChi(df,type="q")
plotChi(df,type="dis")
plotChi(df,type="den")
```

plotDUnif	*Probability Mass and/or Distribution Function Representations associated with a Discrete Uniform Distribution*

Description

`plotDUnif` represents the probability mass and/or the distribution function associated with a Discrete Uniform distribution with support `x`.

Usage

```
plotDUnif(x, type = "b", col = "grey")
```

Arguments

x	support of the discrete variable.
type	a character string giving the type of desired plot. The following values are possible: "b" (default) for probability mass function and distribution function representations together, "d" for distribution function representation and "p" for probability mass function representation.
col	a single colour associated with the probability mass function representation; default to "grey".

Value

A matrix containing the probability mass and the distribution function associated with each point of the support (denoted by x) of a Discrete Uniform distribution.

Examples

```
x=1:5
plotDUnif(x,type="d")
plotDUnif(x,type="p",col="pink")
plotDUnif(x)
```

---

plotExp — *Density Function, Distribution Function and/or Quantile Function Representations associated with a Exponential Distribution*

Description

plotExp represents density, distribution and/or quantile functions associated with a Exponential distribution with certain parameter lambda.

Usage

```
plotExp(lambda, type = "b", col = "black")
```

Arguments

lambda	the parameter of the Exponential distribution (1/mean).
type	a character string giving the type of desired plot. The following values are possible: "b" (default) for density function, distribution function and quantile function representations together, "dis" for distribution function representation, "den" for density function representation and "q" for quantile function representation.
col	a single colour associated with the different representations; default to "black".

Value

This function is called for the side effect of drawing the plot.

Examples

```
lambda=0.5
plotExp(lambda)
plotExp(lambda,col="red")
plotExp(lambda,type="q")
plotExp(lambda,type="dis")
plotExp(lambda,type="den")
```

---

plotFS — *Density Function, Distribution Function and/or Quantile Function Representations associated with a F-Snedecor Distribution*

Description

plotBeta represents density, distribution and/or quantile functions associated with a F-Snedecor distribution with certain df1 and df2 degrees of freedom.

Usage

```
plotFS(df1, df2, type = "b", col = "black")
```

Arguments

df1, df2	the degrees of freedom of the F-Snedecor distribution.
type	a character string giving the type of desired plot. The following values are possible: "b" (default) for density function, distribution function and quantile function representations together, "dis" for distribution function representation, "den" for density function representation and "q" for quantile function representation.
col	a single colour associated with the different representations; default to "black".

Value

This function is called for the side effect of drawing the plot.

Examples

```
df1=10;df2=15
plotFS(df1,df2)
plotFS(df1,df2,col="red")
plotFS(df1,df2,type="q")
plotFS(df1,df2,type="dis")
plotFS(df1,df2,type="den")
```

`plotGamma` *Density Function, Distribution Function and/or Quantile Function Representations associated with a Gamma Distribution*

Description

`plotGamma` represents density, distribution and/or quantile functions associated with a Gamma distribution with certain parameters `lambda` and `shape`.

Usage

```
plotGamma(lambda, shape, type = "b", col = "black")
```

Arguments

`lambda, shape`	parameters of the Gamma distribution (mean equal to `shape/lambda`).
`type`	a character string giving the type of desired plot. The following values are possible: "b" (default) for density function, distribution function and quantile function representations together, "dis" for distribution function representation, "den" for density function representation and "q" for quantile function representation.
`col`	a single colour associated with the different representations; default to "black".

Value

This function is called for the side effect of drawing the plot.

Examples

```
lambda=0.5;shape=4
plotGamma(lambda,shape)
plotGamma(lambda,shape,col="red")
plotGamma(lambda,shape,type="q")
plotGamma(lambda,shape,type="dis")
plotGamma(lambda,shape,type="den")
```

`plotHyper` *Probability Mass and/or Distribution Function Representations associated with a Hypergeometric Distribution*

Description

`plotHyper` represents the probability mass and/or the distribution function associated with a Hypergeometric distribution with parameters `N`, `n` and `k`.

Usage

```
plotHyper(N, n, k, type = "b", col = "grey")
```

Arguments

N	the population size.
n	the number of draws.
k	the number of success states in the population.
type	a character string giving the type of desired plot. The following values are possible: "b" (default) for probability mass function and distribution function representations together, "d" for distribution function representation and "p" for probability mass function representation.
col	a single colour associated with the probability mass function representation; default to "grey".

Value

A matrix containing the probability mass and the distribution function associated with each point of the support of a Hypergeometric distribution with parameters N, n and k.

Examples

```
N=20;n=12;k=5
plotHyper(N,n,k,type="d")
plotHyper(N,n,k,type="p",col="pink")
plotHyper(N,n,k)
```

---

plotNegBinom *Probability Mass and/or Distribution Function Representations associated with a Negative Binomial Distribution*

Description

plotNegBinom represents the probability mass and/or the distribution function associated with a Negative Binomial distribution with certain parameters n and p.

Usage

```
plotNegBinom(n, p, type = "b", col = "grey")
```

Arguments

n	the number of successful Bernoulli trials.
p	the probability of success associated with the Bernoulli trial.
type	a character string giving the type of desired plot. The following values are possible: "b" (default) for probability mass function and distribution function representations together, "d" for distribution function representation and "p" for probability mass function representation.
col	a single colour associated with the probability mass function representation; default to "grey".

Details

Note that if `n=1`, the Negative Binomial distribution is also known as Geometric distribution.

Value

A matrix containing the probability mass and the distribution function associated with each point of the support of a Negative Binomial distribution with parameters `n` and `p`.

Examples

```
n=3;p=0.3
plotNegBinom(n,p,type="d")
plotNegBinom(n,p,type="p",col="pink")
plotNegBinom(n,p)
```

---

plotNorm — *Density Function, Distribution Function and/or Quantile Function Representations associated with a Normal Distribution*

Description

`plotNorm` represents density, distribution and/or quantile functions associated with a Normal distribution with certain parameters `mu` and `sigma`.

Usage

```
plotNorm(mu, sigma, type = "b", col = "black")
```

Arguments

mu	the mean of the Normal distribution.
sigma	the standard deviation of the Normal distribution.

`type` a character string giving the type of desired plot. The following values are possible: "b" (default) for density function, distribution function and quantile function representations together, "dis" for distribution function representation, "den" for density function representation and "q" for quantile function representation.

`col` a single colour associated with the different representations; default to "black".

Value

This function is called for the side effect of drawing the plot.

Examples

```
mu=10; sigma=5
plotNorm(mu,sigma)
plotNorm(mu,sigma,col="red")
plotNorm(mu,sigma,type="q")
plotNorm(mu,sigma,type="dis")
plotNorm(mu,sigma,type="den")
```

`plotPois` *Probability Mass and/or Distribution Function Representations associated with a Poisson Distribution*

Description

`plotPois` represents the probability mass and/or the distribution function associated with a Poisson distribution with parameter `lambda`.

Usage

```
plotPois(lambda, type = "b", col = "grey")
```

Arguments

`lambda` mean of the Poisson distribution.

`type` a character string giving the type of desired plot. The following values are possible: "b" (default) for probability mass function and distribution function representations together, "d" for distribution function representation and "p" for probability mass function representation.

`col` a single colour associated with the probability mass function representation; default to "grey".

Value

A matrix containing the probability mass and the distribution function associated with each point of the support of a Poisson distribution with parameter `lambda`.

Examples

```
lambda=2
plotPois(lambda,type="d")
plotPois(lambda,type="p",col="pink")
plotPois(lambda)
```

---

plotReg *Representation of a Linear Regression Model*

Description

Representation of a Linear Regression Model

Usage

```
plotReg(x, y, main = "Linear Regression Model",
  xlab = "Explanatory variable (X)", ylab = "Response variable (Y)",
  col.points = "black", col.line = "red", pch = 19, lwd = 2,
  legend = TRUE)
```

Arguments

`x`	a numeric vector that contains the values of the explanatory variable.
`y`	a numeric vector that contains the values of the response variable.
`main`	a main title for the plot; default to "Linear Regression Model".
`xlab`	x-axis label; default to "Explanatory variable (X)".
`ylab`	y-axis label; default to "Response variable (Y)".
`col.points`	a single colour associated with the sample points; default to "black".
`col.line`	a single colour associated with the fitted linear regression model; default to "red".
`pch`	an integer specifying a symbol or a single character to be used as the default in plotting points; default to 19.
`lwd`	line width for the estimated model, a positive number; default to 2.
`legend`	logical value; if TRUE (default), a legend with details about fitted model is included.

Value

This function is called for the side effect of drawing the plot.

Examples

```
x=rnorm(100)
error=rnorm(100)
y=1+5*x+error
plotReg(x,y)
```

plotTS — *Density Function, Distribution Function and/or Quantile Function Representations associated with a T-Student Distribution*

Description

plotTS represents density, distribution and/or quantile functions associated with a T-Student distribution with df degrees of freedom.

Usage

```
plotTS(df, type = "b", col = "black")
```

Arguments

df	the degrees of freedom of the T-Student distribution.
type	a character string giving the type of desired plot. The following values are possible: "b" (default) for density function, distribution function and quantile function representations together, "dis" for distribution function representation, "den" for density function representation and "q" for quantile function representation.
col	a single colour associated with the different representations; default to "black".

Value

This function is called for the side effect of drawing the plot.

Examples

```
df=10
plotTS(df)
plotTS(df,col="red")
plotTS(df,type="q")
plotTS(df,type="dis")
plotTS(df,type="den")
```

plotUnif — *Density Function, Distribution Function and/or Quantile Function Representations associated with a Uniform Distribution*

Description

plotUnif represents density, distribution and/or quantile functions associated with a Uniform distribution with min and max the lower and upper limits, respectively.

Usage

```
plotUnif(min, max, type = "b", col = "black")
```

Arguments

min	minimum value of the Uniform distribution.
max	maximum value of the Uniform distribution.
type	a character string giving the type of desired plot. The following values are possible: "b" (default) for density function, distribution function and quantile function representations together, "dis" for distribution function representation, "den" for density function representation and "q" for quantile function representation.
col	a single colour associated with the different representations; default to "black".

Value

This function is called for the side effect of drawing the plot.

Examples

```
min=0 ; max=1
plotUnif(min,max)
plotUnif(min,max,col="red")
plotUnif(min,max,type="q")
plotUnif(min,max,type="dis")
plotUnif(min,max,type="den")
```

---

proportion.CI *Large Sample Confidence Interval for a Population Proportion*

Description

`proportion.CI` provides a pointwise estimation and a confidence interval for a population proportion.

Usage

```
proportion.CI(x, n, conf.level)
```

Arguments

x	a single numeric value corresponding with either the proportion estimate or the number of successes of the sample.
n	a single positive integer corresponding with the sample size.
conf.level	a single numeric value corresponding with the confidence level of the interval; must be a value in (0,1).

Details

Counts of successes and failures must be nonnegative and hence not greater than the corresponding numbers of trials which must be positive. All finite counts should be integers.

If the number of successes are given, then the proportion estimate is computed.

Value

A list containing the following components:

`estimate`	numeric value corresponding with the sample proportion estimate.
`CI`	a numeric vector of length two containing the lower and upper bounds of the confidence interval.

Independently on the user saving those values, the function provides a summary of the result on the console.

Examples

```
#Given the sample proportion estimate
proportion.CI(0.3, 100, conf.level=0.95)

#Given the number of successes
proportion.CI(30,100,conf.level=0.95)
```

`proportion.test` *Large Sample Test for a Population Proportion*

Description

`proportion.test` allows to compute a hypothesis test for a population proportion.

Usage

```
proportion.test(x, n, p0, alternative = "two.sided", alpha = 0.05,
  plot = TRUE, lwd = 1)
```

Arguments

`x`	a positive number indicating the counts of successes or, if number between 0 and 1, probability of success.
`n`	a single positive integer corresponding with the sample size.
`p0`	a positive number in (0,1) corresponding with the proportion to test.
`alternative`	a character string specifying the alternative hypothesis, must be one of "`two.sided`" (default), "`greater`" or "`less`".
`alpha`	a single number in (0,1) corresponding with significance level.
`plot`	a logical value indicating whether to display a graph including the test statistic value for the sample, its distribution, the rejection region and p-value.
`lwd`	a single number indicating the line width of the plot.

Details

Counts of successes and failures must be nonnegative and hence not greater than the corresponding numbers of trials which must be positive. All finite counts should be integers. If the number of successes is given, then the proportion estimate is computed.

Value

A list with class "`lstest`" and "`htest`" containing the following components:

`statistic`	the value of the test statistic.
`parameter`	the sample size `n`.
`p.value`	the p-value of the test.
`estimate`	the sample proportion.
`null.value`	the value of `p0` specified by the null.
`alternative`	a character string describing the alternative.
`method`	a character string indicating the method used.
`data.name`	a character string giving the names of the data.
`alpha`	the significance level.
`dist.name`	a character string indicating the distribution of the test statistic.
`statformula`	a character string with the statistic's formula.
`reject.region`	a character string with the reject region.

Examples

```
x <- rbinom(1, 120, 0.6)
proportion.test(x, 120, 0.5, alternative = "greater")
proportion.test(0.6, 120, 0.5, alternative = "greater")
```

`read.data` *Data Input*

Description

`read.data` allows to read a file and create a data frame from it. Wrapper for different data input functions available, namely `data.table::fread`, `readxl::read_excel`, `haven::read_sas`, `haven::read_sav`, `haven::read_dta` and `readODS::read_ods`. The file extensions supported are: csv, dat, data, dta, ods, RDa, RData, sas7bdat, sav, txt, xls and xlsx.

Usage

```
read.data(name, dec = ".", header = "auto", sheet = 1, ...)
```

Arguments

- `name`: a character string with the name of the file including the file extension from which the data are to be read from.
- `dec`: a character string indicating the decimal separator for txt, csv, dat and data files. If not "." (default) then usually ",".
- `header`: a character string indicating if the first data line contains column names, as in `data.table::fread`. Defaults according to whether every non-empty field on the first data line is type character; if so, or TRUE is supplied, then the first row is considered as the variables names and any empty column names are given a default name.
- `sheet`: the sheet to read for xls, xlsx and ods files. Either a string (the name of a sheet) or an integer (the position of the sheet). If not specified, defaults to the first sheet.
- `...`: Further arguments to be passed to `data.table::fread`, `readxl::read_excel`, `haven::read_sas`, `haven::read_sav`, `haven::read_dta` or `readODS::read_ods`.

Value

A `data.frame` containing the data in the specified file or, if Rdata or Rda, an object of class `"ls_str"`.

Examples

```
data <- data.frame(V1 = 1:5*0.1, V2 = 6:10)
tf <- tempfile("tp", fileext = c(".csv", ".txt", ".RData"))

write.csv(data, tf[1], row.names = FALSE)
read.data(tf[1])                             # csv
write.csv2(data, tf[1], row.names = FALSE)
read.data(tf[1], dec = ",")                  # csv2
write.table(data, tf[2], row.names = FALSE)
read.data(tf[2])                             # txt
save(data, file = tf[3])
read.data(tf[3])                             # RData
```

S2mu *Variance Estimator when the Population Mean is Known*

Description

`S2mu` computes a estimation of the variance, given a sample x with known population mean (denoted by `mu`).

Usage

```
S2mu(x, mu)
```

Arguments

`x`	a numeric vector containing the sample.
`mu`	the population mean.

Details

Given $\{x_1, \ldots, x_n\}$ a sample of a random variable, the variance estimator when the population mean (denoted by μ) is known can be computed as $S_\mu^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$.

Value

A single numerical value corresponding with the variance estimation when the population mean is known.

Examples

```
x=rnorm(20)
S2mu(x,mu=0)
```

`sample.quantile` *Sample Quantiles*

Description

`sample.quantile` computes a estimation of different quantiles, given a sample x, using order statistics.

Usage

```
sample.quantile(x, tau)
```

Arguments

`x`	a numeric vector containing the sample.
`tau`	the quantile(s) of interest, that must be a number(s) in (0,1).

Details

A quantile `tau` determines the proportion of values in a distribution are above or below a certain limit. For instance, given `tau` a number between 0 and 1, the `tau`-quantile splits the sample into tow parts with probabilities `tau` and `(1-tau)`, respectively.

One possible way to calculate the quantile `tau` would be to ordering the sample and taking as the quantile the smallest data in the sample (first of the ordered sample) whose cumulative relative frequency is greater than `tau`. If there is a point in the sample with a cumulative relative frequency equal to `tau`, then the sample quantile will be calculated as the mean between that point and the next one of the ordered sample.

Value

A number or a numeric vector of `tau`-quantile(s).

A numerical value or vector corresponding with the requested sample quantiles.

Examples

```
x=rnorm(20)
sample.quantile(x,tau=0.5)
sample.quantile(x,tau=c(0.25,0.5,0.75))
```

`sample.sd` *Sample Standard Deviation*

Description

`sample.sd` computes the sample standard deviation of a sample `x`.

Usage

```
sample.sd(x)
```

Arguments

`x`	a numeric vector containing the sample.

Details

Given $\{x_1, \ldots, x_n\}$ a sample of a random variable, the sample standard deviation can be computed as $S = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$.

Value

A single numerical value corresponding with the sample standard deviation.

Examples

```
x=rnorm(20)
sample.sd(x)
```

sample.var — *Sample Variance*

Description

`sample.var` computes the sample variance of a sample `x`.

Usage

```
sample.var(x)
```

Arguments

`x`	a numeric vector containing the sample.

Details

Given $\{x_1, \ldots, x_n\}$ a sample of a random variable, the sample variance can be computed as $S^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$.

Value

A single numerical value corresponding with the sample variance.

Examples

```
x=rnorm(20)
sample.var(x)
```

sicri2018 — *SICRI: information system on risk-taking behaviour*

Description

SICRI collects data on health-related behaviours, as long as they do not involve questions that can be considered sensitive or delicate, since with them simple self-declaration is not a way to obtain valid information. However, although the data collected will mainly refer to behaviours, data of another type may be collected for which simple self-declaration is an accommodated mode of measurement.

Usage

```
data(sicri2018)
```

Format

sicri2018 is a data frame with 7853 cases (rows) and 18 variables (columns) selected from the original and complete data base. The selection was done to have an overview of different types of random variables, as well as the topic interest for the students to discuss about.

In what follows the explanation and codification of the different variables is detailed. For every variable, we provide its name on the data base, the explanation of its content and all the possible categories, whether it makes sense, with their codes on the data base between brackets.

sex	men (1); women (2)
age	age in years
weight	weight in kilograms
height	height in centimetres
bmi	body mass index
prov	province: A Coruña (15); Lugo (27); Ourense (32); Pontevedra (36)
employ	employment situation: employed (1); unemployed (2); housework (3); retired (4); student (5); other (6)
education	educational level: no studies (1); basic level (2); medium level (3); upper level (4)
civilstat	civil status: living as a couple (1); not living as a couple (2)
smoke	whether has ever smoked: no (0); yes (1)
Ncigarw	number of cigaretts per week
TimeNS	time (in years) since the last time the person smoked
vac	whether the person trust vaccines: no (0); yes (1); not know (3)
Ndrinks	number of usual glasses drunk, any day the person takes alcoholic drinks
netuse	time (in minutes) surfing the Net in the last four weeks
game	whether the person has spent money on gambling in the last 12 months: no (0); yes (1)
blood	whether the person has ever been a blood donor
can	whether the person has ever smoked cannabis

Source

This data has been collected from SERGAS (Galician Health Service) Database on February 2021 https://www.sergas.es/Saude-publica/SICRI-2018-Microdatos

Examples

```
data(sicri2018)
bmi <- sicri2018$bmi
Histogram(bmi,col="pink")
```

Smu *Standard Deviation Estimator when the Population Mean is Known*

Description

Smu computes a estimation of the standard deviation, given a sample x with known mean (denoted by mu).

Usage

```
Smu(x, mu)
```

Arguments

x	a numeric vector containing the sample.
mu	the population mean.

Details

Given $\{x_1, \ldots, x_n\}$ a sample of a random variable, the standard deviation estimator when the population mean (denoted by μ) is known can be computed as $S_\mu = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$.

Value

A single numerical value corresponding with the standard deviation estimation when the population mean is known.

Examples

```
x=rnorm(20)
Smu(x,mu=0)
```

---

variance.CI *Confidence Interval for the Variance and the Standard Deviation of a Normal Population*

Description

variance.CI provides a pointwise estimation and a confidence interval for the variance and the standard deviation of a normal population in both scenarios: known and unknown population mean.

Usage

```
variance.CI(x = NULL, s = NULL, sc = NULL, smu = NULL, mu = NULL,
  n = NULL, conf.level)
```

Arguments

`x`	a numeric vector containing the sample.
`s`	a single numeric value corresponding with the sample standard deviation.
`sc`	a single numeric value corresponding with the cuasi-standard deviation.
`smu`	if known, a single numeric value corresponding with the estimation of the standard deviation for known population mean.
`mu`	if known, a single numeric value corresponding with the population mean. Even when the user provides smu, mu is still needed.
`n`	a single positive integer corresponding with the sample size; not needed if the sample is provided.
`conf.level`	a single numeric value corresponding with the confidence level of the interval; must be a value in (0,1).

Details

The formula interface is applicable when the user provides the sample and also when the user provides the value of sample characteristics (cuasi-standard deviation or sample standard deviation and the sample size).

Value

A list containing the following components:

`var.estimate`	the cuasi-variance for unknown population mean, and the estimation of the variance for known population mean.
`sd.estimate`	the cuasi-standard deviation for unknown population mean and the estimation of the standard deviation for known population mean.
`CI.var`	a numeric vector of length two containing the lower and upper bounds of the confidence interval for the population variance.
`CI.sd`	a numeric vector of length two containing the lower and upper bounds of the confidence interval for the population standard deviation.

Independently on the user saving those values, the function provides a summary of the result on the console.

Examples

```
#Given the estimation of the standard deviation with known population mean
dat=rnorm(20,mean=2,sd=1)
smu=Smu(dat,mu=2)
variance.CI(smu=smu,mu=2,n=20,conf.level=0.95)

#Given the sample with known population mean
dat=rnorm(20,mean=2,sd=1)
variance.CI(dat,mu=2,conf.level=0.95)

#Given the sample with unknown population mean
```

```
dat=rnorm(20,mean=2,sd=1)
variance.CI(dat,conf.level=0.95)

#Given the cuasi-standard deviation with unknown population mean
dat=rnorm(20,mean=2,sd=1)
variance.CI(sc=sd(dat),n=20,conf.level=0.95)
```

`variance.test` *One Sample Variance Test of a Normal Population*

Description

`variance.test` allows to compute hypothesis tests for the variance of a Normal population in both scenarios: known or unknown population mean.

Usage

```
variance.test(x = NULL, s = NULL, sc = NULL, smu = NULL, mu = NULL,
  n = NULL, sigma02, alternative = "two.sided", alpha = 0.05,
  plot = TRUE, lwd = 1)
```

Arguments

`x`	a numeric vector containing the sample.
`s`	a single numeric value corresponding with the sample standard deviation.
`sc`	a single numeric value corresponding with the cuasi-standard deviation.
`smu`	if known, a single numeric value corresponding with the estimation of the standard deviation for known population mean.
`mu`	if known, a single numeric value corresponding with the population mean. Even when the user provides smu, mu is still needed.
`n`	a single positive integer corresponding with the sample size; not needed if the sample is provided.
`sigma02`	a single number corresponding with the variance to test.
`alternative`	a character string specifying the alternative hypothesis, must be one of "`two.sided`" (default), "`greater`" or "`less`".
`alpha`	a single number in (0,1), significance level.
`plot`	a logical value indicating whether to display a graph including the test statistic value for the sample, its distribution, the rejection region and p-value.
`lwd`	a single number indicating the line width of the plot.

Details

The formula interface is applicable when the user provides the sample or values of the sample characteristics (cuasi-standard deviation or sample standard deviation).

Value

A list with class "lstest" and "htest" containing the following components:

statistic	the value of the test statistic.
parameter	the degrees of freedom of the statistic's distribution.
p.value	the p-value of the test.
estimate	the cuasi-variance.
null.value	the value specified by the null.
alternative	a character string describing the alternative.
method	a character string indicating the method used.
data.name	a character string giving the names of the data.
alpha	the significance level.
dist.name	a character string indicating the distribution of the test statistic.
statformula	a character string with the statistic's formula.
reject.region	a character string with the reject region.
unit	a character string with the units.

Examples

```
x <- rnorm(50, mean = 1, sd = 2)
# unknown population mean
variance.test(x, sigma02 = 3.5)
variance.test(sc = sd(x), n = 50, sigma02 = 3.5)
# known population mean
variance.test(x, sigma02 = 3.5, mu = 1)
smu <- Smu(x, mu = 1)
variance.test(smu = smu, n = 50, sigma02 = 3.5)
```

Index