

Package ‘ResearchMethods’

April 17, 2009

Type Package

Title Using GUIs to help teach statistics to non-statistics students

Version 1.01

Date 2008-12-12

Author Mohamed Abdoell and Sam Stewart

Maintainer Sam Stewart <samstewart11@gmail.com>

Depends tcltk,irr,gplots,ellipse

Description

License GPL

Repository CRAN

Date/Publication 2008-12-14 11:08:16

R topics documented:

agpop	2
BlandAltman	3
bootSequence	5
bootSingle	6
caseRatio	7
cltDemo	8
confInt	9
GUIDemo	10
histSample	10
menuSampleSize	12
MFSV	13
minimAllocGUI	14
multHypothesis	15
myHist	16
outlierTest	17

powerDemo	18
ratioCompare	19
reactGame	20
rhoRange	21
rhoScale	23
rocCurve	24
sampleAssign	26
sampleSchemes	27
sumSquares	28
tau	29

Index	30
--------------	-----------

agpop

Farming Data

Description

The results from the US 1992 Census of Agriculture

Arguments

agpop	A dataset with 3078 entries and 15 variables:
\$county	County name
\$state	state abbreviation
\$acres92	Number of acres devoted to farms in 1992 (-99=missing)
acres87	Number of acres devoted to farms in 1987 (-99=missing)
acres82	Number of acres devoted to farms in 1982 (-99=missing)
farms92	Number of farms in 1992 (-99=missing)
farms87	Number of farms in 1987 (-99=missing)
farms82	Number of farms in 1982 (-99=missing)
largef92	Number of farms, with 100 acres or more, in 1992 (-99=missing)
largef87	Number of farms, with 100 acres or more, in 1987 (-99=missing)
largef82	Number of farms, with 100 acres or more, in 1982 (-99=missing)
smallf92	Number of farms, with 9 acres or less, in 1992 (-99=missing)
smallf87	Number of farms, with 9 acres or less, in 1987 (-99=missing)
smallf82	Number of farms, with 9 acres or less, in 1982 (-99=missing)
region	S=South, W=west, NC=north central, and NE=northeast

Details

As stated above, a -99 entry indicates a missing value. To replace the missing values with NA, use the example code below.

See Also[MFSV](#)**Examples**

```
#to call the dataset
data("agpop") #note the quotation marks

#to convert the -99's to NA's
library(gdata)
agpop2 = unknownToNA(agpop, unknown=-99) #it is preferable to save new dataset under
```

BlandAltman

*BlandAltman Plot***Description**

Using a graphical user interface (GUI) this function performs a Bland Altman plot, and allows for manipulation of the variables within the plot.

Usage

```
BlandAltman(x, y, gui=TRUE, bandsOn=FALSE, biasOn=FALSE, regionOn=FALSE, smooth=FAI
```

Arguments

x	The observations from the old measurement technique.
y	The observations from the new measurement technique.
gui	An indicator of whether the interface should be activated.
bandsOn	An indicator of whether the confidence bands should be plotted.
biasOn	An indicator of whether the bias region should be plotted.
regionOn	An indicator of whether the region of agreement should be plotted.
smooth	An indicator of whether lines of best fit (linear and non-linear) should be included in the plot.
sig	Initial value of the scalar on sigma

Details

The Bland-Altman plot is a visual tool for comparing two different methods of measuring the same value, when the true value being measured is not known. The purpose of a Bland Altman plot is to try and determine whether a new method of measurement is better than an established one, using a hypothesis testing approach. See the referenced papers for more information.

This function produces one or two windows. The Tk window, or the control window, is only produced when `gui=T`, and provides controls to manipulate the plot in real time. The control window contains three check buttons and a slider bar, which provide the manipulation of the Bland

Altman plot. The top button adds the confidence bounds, the second adds the bias line, and the third adds the region of agreement. Note that the bias line and region of agreement will not work unless the confidence bounds are on. The slider bar allows for manipulation of the confidence bounds: the standard confidence region is $2 \times \text{sigma}$, but this is strictly a statistical standard, and there may be a desire to widen or shrink the confidence region given the specific data. The slider bar has a range of $[0.1, 5]$ within which the multiplier on sigma may be set.

The second window is the plotting window, featuring a scatter plot of the Bland Altman data. See the referenced Bland Altman paper for a more detailed explanation of the Bland Altman process: a brief explanation is that the points are plotted with the difference between two observations on the y-axis, and the mean of the two observations on the x-axis. The confidence bounds, when added, are plotted as dotted red lines, and all points within the confidence bounds are coloured green, all points outside the confidence bounds in red. The bias region, plotted as an orange field, is the range between the difference = 0 line and the average difference, and the region of agreement is merely a green shading of the area within the confidence bounds. The lines of best fit are regression and lowess fits for the linear and non-linear lines respectively.

Value

No meaningful value is returned, this function is run only for its plotting functions. The variable `BAenv` is left behind so the variables used in the plotting function may be manipulated.

Note

The slider has a step size of .1: the value of the slider may be manipulated by either clicking and dragging the slider or clicking and holding the space to the left/right of the slider, which will decrease/increase the slider by the step size.

The function was designed to work with the GUI. The ability to plot without it was added to allow the function to be embedded into other programs such as Sweave. Whenever possible, it is better to use the GUI controls.

Author(s)

Mohamed Abdoell <mohamed.abodolell@dal.ca> and Sam Stewart <samstewart11@gmail.com>

References

Altman, Douglas G. and Bland, J. Martin. "Statistical Methods For Assessing Agreement Between Two Methods of Clinical Measurement". *Lancet*, 1986, i:307-310.

See Also

[MFSV](#)

Examples

```
#a simple example
data("MFSV")
BlandAltman(MFSV$MF, MFSV$SV)

# This will only produce a BlandAltman Plot with the region of agreement added
```

```
BlandAltman(MFSV$MF, MFSV$SV, gui=FALSE, bandsOn=TRUE, regionOn=TRUE)

#this example does not make sense in the context of the function
#but does provide a dataset with a large bias-value
data("agpop")
BlandAltman(agpop$farms87, agpop$farms92)
```

bootSequence	<i>A demonstration of how bootstrapping works, taking multiple bootstrap samples and watching how the means of those samples begin to normalize.</i>
--------------	--

Description

This function uses a graphical user interface (GUI) to demonstrate how bootstrapping works, by bootstrapping from a submitted dataset sequentially and watching the sampled means move towards a normal distribution. The confidence intervals also narrow.

Usage

```
bootSequence(dat)
```

Arguments

dat the dataset to sample from.

Details

This function produces two windows. The Tk window, or the control window, contains three sampling buttons: the top button takes a single bootstrap sample, and the bottom two buttons take 10 and 100 bootstrap samples.

The second window is a visual demonstration of bootstrapping. There are three windows: the top-left window is the histogram of the submitted dataset. The top right window is a histogram of the most recent bootstrap sample (for the 10 and 100 size samples, this plot is blank), and the bottom plot is a histogram of the means of the sample.

The histogram at the bottom has the 95% confidence intervals, calculated as the 2.5% and 97.5% quantiles of the bootstrapped means. The blocks outside the confidence interval are coloured red.

Value

No value is returned.

Author(s)

Mohamed Abdoell mohamed.abdoell@dal.ca and Sam Stewart samstewart11@gmail.com

References

This function was designed for a course by Mohamed Abdoell

See Also

[bootSingle](#)

Examples

```
data (agpop)
bootSequence (agpop$ farms92)
data (MFSV)
bootSequence (MFSV$MF)
```

bootSingle	<i>A demonstration of how bootstrapping works step by step for one function.</i>
------------	--

Description

This function uses a graphical user interface (GUI) to demonstrate how bootstrapping works, by bootstrapping a sample of the alphabet, either step-by-step or as an entire sample.

Usage

```
bootSingle ()
```

Details

This function produces two windows. The Tk window, or the control window, contains two sampling button: the top button takes an entire sample, the bottom button takes a stepwise sample: note that the first 26 clicks take the samples and then the button changes so that it can sort the sample.

The second window is a visual demonstration of bootstrapping: the top is a color-coded bootstrapped sample, while the bottom is the population sampled from: when a circle in the bottom turns red it means that it has been sampled from.

Value

No value is returned.

Author(s)

Mohamed Abdoell <mohamed.abdoell@dal.ca> and Sam Stewart <samstewart11@gmail.com>

References

This function was designed for a course by Mohamed Abdoell

See Also[bootSequence](#)**Examples**

```
bootSingle()
```

`caseRatio`*The effect of case:control ratios on power*

Description

Using a Graphical User Interface (GUI), this function demonstrates the effect of increasing the proportion of patients allocated to the test treatment in a case:control study.

Usage

```
caseRatio(GUI=TRUE, alpha=0.05, beta=0.1)
```

Arguments

<code>GUI</code>	A boolean determining allowing the GUI to be removed from the function, allowing the plot to be embedded into other code.
<code>alpha</code>	The significance level for the experiment.
<code>beta</code>	The initial beta value for the experiment.

Details

This function produces one or two windows. The optional window is the control window, which allows the user to set the alpha and beta values, and manipulate in real time what proportion of the patients are allocated to the treatment option, rather than the control option. The second window is the plotting window, in which the curve is the possible change in power as the proportion allocated to the treatment increases. The point at which the proportion is currently set is indicated by the two red lines.

Value

No value is returned by this function, although the function environment, `CRenv`, is left so that the values may be retrieved.

Author(s)

Mohamed Abdoell mohamed.abdoell@dal.ca and Sam Stewart samstewart11@gmail.com

References

Pocock, S. J. Allocation of patients to treatment in clinical trials. *Biometrika*, 1979. 35, 183-197.

Examples

```
caseRatio()  
#without the GUI  
caseRatio(GUI=FALSE)  
#with new initial values  
caseRatio(alpha=0.01,beta=0.2)
```

cltDemo

A demonstration of how the central limit theorem works on samples of size n.

Description

This function demonstrates how the central theorem works on a variety of distributions, by allowing the user to manipulate the size of a sample and see that, if the sample size is large enough, the distribution of the sample mean is approximately normal.

Usage

```
cltDemo()
```

Details

This function produces two windows. The Tk window, or the control window, A set of buttons to select the distribution to sample from, as well as a button to increment the sample size and an entry box to set the value of the desired parameter for the exponential and chi-squared distributions.

The plotting window has two figures: the one on the left is a histogram of the means from 10,000 samples of size n from the chosen distribution. The figure on the right is a Q-Q plot comparing the quantiles of the sample means with the quantiles of the normal distribution.

Value

No value is returned.

Author(s)

Kevin Thorpe, Mohamed Abdolell <mohamed.abdolell@dal.ca> and Sam Stewart <samstewart11@gmail.com>

Examples

```
cltDemo()
```

Description

Using a Graphical User Interface (GUI), this function demonstrates what exactly a confidence means, and how random samples will fall inside and outside of it.

Usage

```
confInt(...)
```

Arguments

... Additional arguments...

Details

This function produces two windows: a Tk window to manipulate the variables, and a plotting window that plots the confidence intervals. The TK window has controls to manipulate the mean and standard deviation of the normal distribution to be sampled from, as well as a slider to set the value of alpha. It also has two entry boxes in which n, the size of the sample, and reps, the number of samples, can be set. Note that changing any of the variables except alpha requires a re-sampling of the data, and will therefore completely change the plot.

The second window is the plotting window. In it the y-axis represents the sample, and the x-axis represents the repetitions. Each point represents a mean of a sample, with the lines about it being it's own confidence interval. The green lines represent the confidence interval given the parameters of the plot. If a mean falls outside the green confidence intervals, it is marked in red.

Value

No meaningful value is returned, though the computational environment is retain in CIenv to allow the user to check some of the values plotted.

Note

This function relies on the `plotCI` function in the `gplots` library. To see a non-interactive version of the function, look at `ci.examp` in the `TeachingDemos` library.

Author(s)

Mohamed Abdoell <mohamed.abdoell@dal.ca> and Sam Stewart <samstewart11@gmail.com>

Description

Using Tcl/Tk to create Graphical User Interfaces (GUIs), this package is designed to provide real time controls to basic statistical plotting functions in order to help with the teaching of these methods. The package was originally designed to accompany a research methods course for the Radiology Department at Dalhousie University.

Details

Package: GUIDemo
Type: Package
Version: 1.0
Date: 2007-07-10
License: What license is it under?

The functions in the package take very few variables, and are most often implemented using the form `function(x, y)` or `function(dat)` where `x` and `y` are vectors, and `dat` is a matrix, most often $n \times 2$, or a vector. Calling the function initiates two windows: a plotting window and a Tk control window, which will manipulate the corresponding plot.

The function depends heavily on the Tcl/Tk library in R, which is poorly documented and depends heavily on the Tcl/Tk language itself. Because of this, how the boxes appear and how they function is not guaranteed to work exactly the same on every machine. Care has been taken to test the products on three different operating systems, Windows, Mac OS and Ubuntu Linux, but the results are not guaranteed. Any problems should first be checked against the Tcl/Tk documentation, and then directed to the contact person below.

Author(s)

Mohamed Abdolell <mohamed.abdolell@dal.ca> and Sam Stewart

Maintainer: Sam Stewart <samstewart11@gmail.com>

References

"Tk Commands Manual". <http://www.tcl.tk/man/tcl8.4/TkCmd/contents.htm>. Last accessed July 17, 2007.

Description

Using a graphical user interface (GUI) this function demonstrates that, for a variety of distributions, as the sample size increases, the the sample-histogram converges to the true distribution.

Usage

```
histSample(dist=NA, gui=TRUE, yMax=FALSE, n=10000, sub=NaN, par=c(0, 1))
```

Arguments

<code>dist</code>	the distribution being used, see below for a list.
<code>gui</code>	An indicator of whether the interface should be activated.
<code>yMax</code>	Boolean dictating whether the bounds should be (0,1), FALSE, or should be set to fit the data, TRUE
<code>n</code>	The size of the population being sampled
<code>sub</code>	(Optional) number of elements in the sample
<code>par</code>	A scalar or vector containing the required parameters for each of the distributions

Details

This function produces one or two windows. The Tk window, or the control window, is only produced when `gui=T`. It contains a slider that controls how big a sample is being taken from the population. The range of the slider is [1,n]

The second window is a plotting window, featuring a histogram overlain with a line plot. The line plot is the true value of the distribution, and the histogram for the sample defined by the control window.

There are several distributions currently supported by the function. They are as follows (par indicates how there paramters should be entered).

"norm": par = c(mu,sig)

"exp": par=lambda

"chisq": par=c(df,ncp)

"gamma": par=c(shape,scale)

"pois": par=lambda

Value

No meaningful value is returned, this function is run only for its plotting functions. The variable `H$env` is left behind so the variables used in the plotting function may be manipulated.

Note

The slider has a step size of 1: the value of the slider may be manipulated by either clicking and dragging the slider or clicking and holding the space to the left/right of the slider, which will decrease/increase the slider by the step size.

The function was designed to work with the GUI. The ability to plot without it was added to allow the function to be embedded into other programs such as Sweave. Whenever possible, it is better to use the GUI controls.

Author(s)

Mohamed Abdoell <mohamed.abdoell@dal.ca> and Sam Stewart <samstewart11@gmail.com>

References

This plot was designed for a course by Mohamed Abdoell

See Also

[rhoRange](#)

Examples

```
histSample() # a normal mu = 0, sig = 1 sample
histSample('norm',n=500,par=c(0,1)) # a normal mu = 0, sig = 1 sample again
histSample('gamma',n=5000,par=c(3,2)) # a gamma distn, a = 3, b = 2

#Producing just plots
histSample(dist='norm',n=1000,par=c(10,5),gui=FALSE,sub=45)
```

menuSampleSize *Sample Size Calculations*

Description

This function calculates required sample sizes for a variety of data types, and allows for the interactive manipulation of these calculations.

Usage

```
menuSampleSize(...)
```

Arguments

```
...                    Additional arguments...
```

Details

Using a GUI, this function calculates the sample size required for a variety of sampling designs. See the file `functionList.pdf` for a detailed description of the equations.

Note

Because of the way R libraries are compiled, greek characters and subscripts could not be built into the function. To convert the menu to greek characters with subscripts, open the file `menuSampleSize.R` and uncomment the 11 commented lines at the beginning of the file. Note that THIS IS NOT NECESSARY for the program to work correctly, but it does make the menu clearer and easier to relate to the supporting document, `functionList.pdf`.

Author(s)

Mohamed Abdoell <mohamed.abdoell@dal.ca> and Sam Stewart <samstewart11@gmail.com>

References

[../doc/functionList.pdf](#)

MFSV

Doppler vs cross-sectional echocardiography

Description

The results of a study by Zhang et al. (1986), measuring transmitral volumetric flow (MF) by Doppler echocardiography and left ventricular stroke volume (SV) by cross-section echocardiography in 21 patients without aortic valve disease.

Arguments

MFSV	A dataset with 21 entries and 2 variables:
\$MF	transmitral volumetric flow measured by Doppler echocardiography.
\$SV	left ventricular stroke volume measured by cross-sectional echocardiography.

Details

A dataset in which both methods are measuring the same flow rate and the true flow rate is unknown. This dataset is designed to be analyzed using the Bland Altman plot.

See Also

[BlandAltman](#), [agpop](#)

Description

Using a Graphical User Interface, this function is used to allocate patients to treatment groups. It can also write the results of the current allocation to a file, to be re-used at a later point. The purpose of this function is to provide researchers with a simple, effective patient allocation program.

Usage

```
minimAllocGUI (factors=NULL, file=NULL, prob=1)
```

Arguments

<code>factors</code>	an optional list object describing the factors by which the data should be stratified. The list should have as its first element a vector of the names of the different factors, and for each factor, a subsequent list of levels. In the examples, the object <code>f</code> is an example of the factors object.
<code>file</code>	an optional file that contains the data from a previous allocation. If no file is input then the output is written to <code>tempFile.dat</code> ; with an input file, the data is written back to that file, with a backup of it being saved in <code>tempFile.dat</code> .
<code>prob</code>	this is the probability that the new subject will be allocated to the group that has fewer patients. Pocock's design had the probability set to 1, but the problem has evolved to include a probability so that the allocation of the next patient cannot be predicted.

Details

This function produces two or three windows, depending on what the input is. The first, and optional, window is a tcltk control window designed to allow the user to set the factors by which the patients will be stratified. It contains two input lines: the first allows the user to set the name of the first factor, the second allows the user to set the levels of the named factor. Pressing `continue` adds the current factor to the list of factors, pressing `Return` `Factors` returns the list of factors WITHOUT the one in the input lines. This step may be skipped by including a factor list, such as `f` in the examples.

Once the factors have been set, two more windows pop up. The first is a tcltk control window that allows the user to set all the strata variables, and then allocate the patient. Note that, if two buttons are checked in the same strata then both become unchecked, in order to avoid costly errors. There are three buttons in this window: `Allocate` assigns a patient with the given attributes to a group. The other two buttons allow the user to quit with or without saving the new results. Note that quitting the program without using the buttons is equivalent to quitting without saving.

The final window is the display of the allocation. The left column is a list of the factors, with their levels in the second column, and then how many patients are allocated to each group.

Value

No formal values are returned, though a file is saved with the relevant data, either to `tempFile.dat` or the input file.

Warning

Because of the way the function is programmed, YOU CANNOT USE SEMICOLONS (;) in the names of the factors or the factor levels.

Author(s)

Mohamed Abdoell <mohamed.abdoell@dal.ca> and Sam Stewart <samstewart11@gmail.com>

References

Pocock, Stuart, J. "Clinical Trials: A practical Approach". John Wiley and Sons, Toronto: 1988.

Examples

```
# the most basic way to call the function is without params
minimAllocGUI()

# a list of factors can be used in the following way
f = list()
f[1][[1]] = c("Performance Status", "Age", "Disease-free interval", "Dominant metastatic lesion")
f[2][[1]] = c("Ambulatory", "Non-ambulatory")
f[3][[1]] = c("<50", ">=50")
f[4][[1]] = c("<2 years", ">=2 years")
f[5][[1]] = c("Visceral", "Osseous", "Soft Tissue")

minimAllocGUI(f)

# once a file has been saved, it may be recalled
## Not run: minimAllocGUI(file="tempFile.dat")
```

multHypothesis *The effect of multiple hypothesis testing*

Description

Using a tcltk Graphical User Interface (GUI), this function demonstrates the potential dangers involved in testing multiple hypotheses with dependent data.

Usage

```
multHypothesis(GUI=TRUE, alpha=0.05, hyp=1)
```

Arguments

GUI	A boolean value that lets the user turn the control window off, to allow the function to be embedded in other code.
alpha	The initial significance level.
hyp	The initial number of hypotheses being tested.

Details

The function creates one or two windows. The optional GUI window allows the user to set the significance level, alpha, and to change in real time the number of hypotheses being tested, k. The other window is a plot of the potential chance of a false rejection of the null hypothesis, with indicators marked in red of the exact probability at the current number of tests.

Value

No tangible value is returned, though the environment `MHenv` is left so the user can access the variables.

Author(s)

Mohamed Abdoell mohamed.abdoell@dal.ca and Sam Stewart samstewart11@gmail.com

Examples

```
# The simplest way to run the function, with the GUI active
multHypothesis()

#running it without a GUI
multHypothesis(GUI=FALSE)

#Starting with initial values
multHypothesis(alpha=0.1,hyp=10)
```

myHist

Drawing simple histograms within a plotting area

Description

This function is an internal function used to draw histograms within a plotting area.

Usage

```
myHist(xl, yb, xr, yt, hDat, breaks = NULL, sample = TRUE, CIs = FALSE, xlab = FA
```

Arguments

xl, yb, xr, yt
hDat
breaks
sample
CIs
xlab
label.bars
right

Author(s)

Mohamed Abdoell <mohamed.abdoell@dal.ca> and Sam Stewart <samstewart11@gmail.com>

References

This plot was designed for a course by Mohamed Abdoell

outlierTest *Effects of removing outliers on regression fits*

Description

Using the `identify` function in R, this function demonstrates the effects that outliers can have on regression by allowing them to be removed in real time.

Usage

```
outlierTest(y, x, output=FALSE)
```

Arguments

y	The response variable for the linear regression
x	The predictor variable for the linear regression
output	a boolean deciding whether to print a summary of each new model to the R terminal

Details

This function takes the input vectors `y` and `x` and performs a simple linear regression using the code `lm(y~x)`. Then, using the `locator()` function it allows the user to point and click on the plot to add or remove modifiers from the dataset. Data points in the model will be colored blue, points excluded will be colored green, and new points will be colored red.

Note

The function breaks down once the dataset is reduced to one point.

Author(s)

Mohamed Abdoell <mohamed.abdoell@dal.ca and Sam Stewart <samstewart11@gmail.com>

References

This plot was designed for a course by Mohamed Abdoell

See Also

[identify](#)

Examples

```
## Not run:

data(MFSV)
outlierTest(MFSV[,1],MFSV[,2])           # simplest test
outlierTest(MFSV[,2],MFSV[,2],output=TRUE) # see the details of each successive model
## End(Not run)
```

powerDemo

A demonstration of the ways in which the power of a study can be adjusted

Description

This function attaches a graphical user interface (GUI) to the power.examp function in the TeachingDemos library, as well as normalizing the two curves (where the power.examp function doesn't).

Usage

```
powerDemo(n=1, sd=1, diff=1, alpha=0.05, xmin=-4, xmax=4, colored=NULL)
```

Arguments

n	The starting value for n
sd	The starting value for stdev
diff	The starting value for diff
alpha	The starting value for alpha
xmin, xmax	The limits for the plotting axes
colored	The name of the variable to color red

Details

This function produces two windows. The Tk window, or the control window, contains four sliders that control the four manipulable values.

The second window is the two plots: the top plot is the distribution if the null hypothesis is true (with the type I error highlighted), the bottom is the distribution if the alternative hypothesis is true (with 1-type II error highlighted). Note that both distributions are normalized.

Value

No value is returned.

Note

The sliders have a step size of 1: the value of a slider may be manipulated by either clicking and dragging the slider or clicking and holding the space to the left/right of the slider, which will decrease/increase the slider by the step size.

Author(s)

Mohamed Abdoell <mohamed.abdoell@dal.ca> and Sam Stewart <samstewart11@gmail.com>

References

This function was designed for a course by Mohamed Abdoell

See Also

`power.examp` from the TeachingDemos library

Examples

```
powerDemo()  
powerDemo(n=100, sd=10, diff=3, alpha=0.025, colored='diff')
```

`ratioCompare`*A demonstration of the approximation of RR by OR*

Description

Using a Graphical User Interface (GUI), this function demonstrates how an odds ratio can be an accurate approximation of relative risk when the probability is low.

Usage

```
ratioCompare()
```

Details

This function is rather straight forward, as it takes no arguments, and the corresponding control window only has two bars on it. The idea is to demonstrate how, as both p_1 and p_2 approach 0, the odds ratio becomes a better approximation of the relative risk.

The only thing of note on the plot is that the logs of the OR and the RR are plotted, rather than the values themselves. This is because the range of both values is $(0, \text{Inf})$, so there is a problem of comparing them when the both approach 0. To see this, set $p_1=0.05$ and $p_2=0.95$. In this situation, the OR and the RR are 0.0028 and 0.0526 respectively, which is a large difference in terms of scale, but is not evident if their differences are taken at face value.

Value

No value is returned, though the environment, `RCenv`, is left for the user to examine.

Author(s)

Mohamed Abdolell <mohamed.abdolell@dal.ca> and Sam Stewart <samstewart11@gmail.com>

reactGame

A game to create data for two-way Anovas'

Description

Using a graphical user interface (GUI) this function demonstrates the effects of changing the range of a dataset on the calculation of correlation, ρ .

Usage

```
reactGame ()
```

Details

This game allows the user to create a dataset of reaction times that can be analyzed using 2-way Anova.

The function produces two windows, a tcltk control window and then the game window in the plotting area.

The plotting window contains a number of options: first there is a space to enter an ID. This can be a number or a character, and is used to track participants in the final analysis. There are two sets of buttons to select whether you are going to click on red or green targets and whether the background should be white or should be colored a similar color to the target. Finally there is an option of either practice, or to save your results to a file. If the named file already exists then the results will be appended to it.

Once the "Play Game!" button is pressed then the plotting window pops up, prompting the user to get ready. Clicking on this screen once starts the game. Once into the game the objective is to click on the screen as soon as you see a target (DO NOT CLICK BEFOREHAND, the resulting data will not make any sense). This will iterate 10 times, after which, if the results were saved, the ANOVA

results will be reported. Note that these results can be unpredictable until there is at least one set observations for each of the four combinations.

Warning

This function does not seem to work with the Quartz window manager on the Mac, and has not been tested on Windows, so the function call should be preceded with the command `X11()` to use the X11 window manager instead.

Author(s)

Mohamed Abdoell <mohamed.abdoell@dal.ca> and Sam Stewart <samstewart11@gmail.com>

References

This plot was designed for a course by Mohamed Abdoell

rhoRange	<i>Size effects on rho calculations</i>
----------	---

Description

Using a graphical user interface (GUI) this function demonstrates the effects of changing the range of a dataset on the calculation of correlation, rho.

Usage

```
rhoRange(x, y, gui=TRUE, xMin=NaN, xMax=NaN, perc=NaN)
```

Arguments

<code>x, y</code>	The two variables to calculate the correlations from, used for the x and y axes respectively
<code>gui</code>	An indicator of whether the interface should be activated.
<code>xMin</code>	(Optional) Specify the lowest x-value in the subset.
<code>xMax</code>	(Optional) Specify the highest x-value in the subset.
<code>perc</code>	(Optional) Specify the proportion about the mean to be included in the subset.

Details

The purpose of this function is to demonstrate the effect of sample size on the calculation of rho. With any of the options set, the program sets the subset to be roughly half the dataset, about the mean. The effect of setting either of the end points is obvious, but note some of the problems in the Warnings.

This function produces two windows. The Tk window, or the control window, is only produced when `gui=T`, and provides controls for manipulating the plot in real time. It contains three slider

bars, which provide two different ways to change the range of the data. The `Lowest` and `Highest` sliders let the user set the lowest and highest values on the x-axis within which the new rho value will be calculated. The `proportion of points` slider allows the user to select what proportion of the points (centered at the mean) should be included in the new calculations. Note that changing the proportion slider moves the `Lowest` and `Highest` sliders, but the opposite is, for now, not true. Note also that, if the `Lowest` and `Highest` sliders are moved, and then the `proportion of points` slider is selected, the other two sliders will reset to the points corresponding to the current proportion.

The second window is a plotting window, featuring a scatterplot and two lines of best fit. The red points, and the corresponding best fit line, are the points that fall within the selection range, manipulated in the Tk window. With this line is a changing rho value, which corresponds to the correlation of the given subset. The green points are the points in the dataset outside the selection window, which means that the green points combined with the red points form the entire dataset, and therefore the green line and the corresponding correlation are on the entire dataset.

Value

No meaningful value is returned, this function is run only for its plotting functions. The variable `RRenv` is left behind so the variables used in the plotting function may be manipulated.

Warning

Because it works in realtime, the function does not deal with large datasets well, ex, with the `agrop` dataset. It may be more useful when working with datasets like these to take a random subset of the data. For an example, see below

The effect of setting any of the optional variables is different if the GUI is on or off. Specifically, setting `xMin` and `xMax` with GUI on does not have much effect, because the GUI is defined by the percent slider. Also note that setting all three variables (which is contradictory to begin with) has two different effects: with the GUI on, the `perc` variable dominates, but with the GUI off, the `xMin` and `xMax` variables dominate. Because of these differences, it is best to only set one of the two options.

Note

This program was written on Ubuntu Linux 7.0.4. Though it should work on all operating systems, because of its use of Tcl/Tk it is only guaranteed on Linux. Any problems with the GUI should be checked against the Tcl/Tk documentation.

The function was designed to work with the GUI. The ability to plot without the GUI on and to change the initial values was added to allow the program to be embedded into programs such as Sweave. Whenever possible, it is better to use the GUI controls.

Author(s)

Mohamed Abdoell <mohamed.abdoell@dal.ca> and Sam Stewart <samstewart11@gmail.com>

References

This plot was designed for a course by Mohamed Abdoell

See Also[rhoScale](#)**Examples**

```
# a simple example
data("MFSV")
rhoRange(MFSV$MF, MFSV$SV)

#These two will produce only the plot
rhoRange(MFSV$MF, MFSV$SV, gui=FALSE, xMin=20, xMax=75)
rhoRange(MFSV$MF, MFSV$SV, gui=FALSE, perc=0.75)

# working with a big dataset
data("agpop")
rhoRange(agpop[, 6], agpop[, 7]) #this will work, but will respond in real ti
```

rhoScale

*Scaling and Shifting rho calculations***Description**

Using a graphical user interface (GUI), this function demonstrates the effect of scaling and shifting a dataset has on its correlation, rho, and its inter-class correlation, icc, on another dataset.

Usage

```
rhoScale(x, y, gui=TRUE, sc=1, sh=0, xlab='x', ylab='y')
```

Arguments

<code>x, y</code>	The two variables from which the correlation is calculated, used as the x and y axes respectively in the plot.
<code>gui</code>	An indicator of whether the interface should be activated.
<code>sc</code>	The initial scaling value.
<code>sh</code>	The initial shifting value.
<code>xlab, ylab</code>	What to label the plots with.

Details

This function produces one or two windows. The Tk window, which only appears if `gui=T` produces two sliders to manipulate the plot window. `Scale` has a range of $[0.1, 5]$ and is the multiplier on the x-variable, labeled *a* (see below). `Shift` has a range of 25% of the data range, about the mean, and is the shift on the x-variable, labeled *b* (see below).

The second window is a plotting window, featuring two scatterplots. The first, the black plot, is of the original data, of the form $y = x$, with a line of best fit through it. The second, the red plot, is the data shifted and scaled, of the form $Y = aX + b$.

Value

No meaningful value is returned, this function is run only for its plotting functions. The variable `RSenv` is left behind so the variables used in the plotting function may be manipulated.

Note

This program was written on Ubuntu Linux 7.0.4. Though it should work on all operating systems, because of its use of Tcl/Tk it is only guaranteed on Linux. Any problems with the GUI should be checked against the Tcl/Tk documentation.

The function was designed to work with the GUI. The ability to plot without it was added to allow the function to be embedded into other programs such as Sweave. Whenever possible, it is better to use the GUI controls.

Author(s)

Mohamed Abdoell <mohamed.abdoell@dal.ca> and Sam Stewart <samstewart11@gmail.com>

References

This plot was designed for a course by Mohamed Abdoell

See Also

[rhoRange](#)

Examples

```
# two simple examples
data("MFSV")
rhoScale(MFSV$MF, MFSV$SV)
data("agpop")
rhoScale(agpop[, 6], agpop[, 7])

# working without the GUI
rhoScale(MFSV$MF, MFSV$SV, gui=FALSE, sc=2, sh=25)
```

rocCurve

A GUI based classification function using a ROC curve

Description

Using a Graphical User Interface (GUI), this function is used to calculate the best classification of data using a ROC curve, and displays the results in ways that demonstrate how a ROC curve works.

Usage

```
rocCurve(x = rnorm(25, 10, 1), y = rnorm(25, 11, 2))
```

Arguments

x	The data who's true value is to be classified as 0.
y	The data who's true value is to be classified as 1.

Details

The ROC curve is used to evaluate the classification of numerical values as belonging to one of two distinct factors. In this function, the factors are represented as 1 and 0. Note that the function only works if the 0 values are in the x-vector and the 1 values are in the y vector.

The function creates two windows. The first window is the control window, which allows the user to change the value of the cutoff, ie, allows the user to set the point above which all values are classified as 1 (y) and below which all values are classified as 0 (x).

The second window contains three plots. The plot in the top left corner is the ROC curve itself. The points indicate indicies at which the sensitivity and specificity change. For more information on how a ROC curve works, consult the references. The red point on the plot is the current cutoff, and the purple line points to the best cutoff point found.

The plot in the top right corner is a simple classification table. Reading from top left to bottom right, the numbers represent: classified correctly as x, classified incorrectly as y, classified incorrectly as x, classified correctly as y.

The final plot at the bottom is a plot of the data points themselves. The points on the bottom are the x values and the points on the top are the y's. The green line represents the current cutoff point, and the purple line indicates the best cutoff point.

Value

This function does not return a value, but leaves the operating environment, RCenv, to allow the user access to the data.

Author(s)

Mohamed Abdolell <mohamed.abdolell@dal.ca> and Sam Stewart <samstewart11@gmail.com>

References

Rosner, Bernard. Fundamentals of Biostatistics, 6th Ed. Thompson Nelson: Toronto, 2006.

Examples

```
# If the function is run without any parameters, then it creates dummy vectors
rocCurve()
# Assuming x and y are the appropriate vectors to be classified
## Not run: rocCurve(x,y)
# Note that rocCurve(y,x) will work very poorly.
```

sampleAssign *A demonstration of different patient allocation methods*

Description

Using a Graphical User Interface (GUI), this function demonstrates to the user different ways to allocate patients to different treatments. The methods are taken from Pocock's book referenced below.

Usage

```
sampleAssign(gui = T, n = 30, treat = 3, coinProb, type = 0, ...)
```

Arguments

gui	An indicator of whether the interface should be activated
n	Number of patient to allocate
treat	Number of treatments to allocate patients to
coinProb	The probability for the biased coin
type	The type of allocation to use: 0=simple allocation, 1=random permuted blocks, 2=Biased Coin
...	Additional arguements...

Details

This function produces two windows. The first window is the control window, from which the user can select one of the following three allocation methods: *Simple Randomization*, *Random Permuted Blocks*, *Biased Coin*. For each of the methods, the number of patients being allocated, *n* can be set, and for the first two, the number of treatments can also be set. Note that Biased Coin can only take two treatments.

The second window is the plotting window, which displays what treatment each patient is allocated to. The first patient is in the top-right corner, and the patients read across as normal. The color coding for the first two methods is self explanatory. For the Biased Coin method, treatment A is colored as red and B as blue. The light red and light blue indicate that the patient is allocated to the same treatment, but they are allocated there because of the biased coin, ie, the light red patients would be blue if a simple randomization scheme were used.

Value

No value is returned, though the environment, SAenv, is left for the user to examine.

Author(s)

Mohamed Abdolell <mohamed.abdolell@dal.ca> and Sam Stewart <samstewart11@gmail.com>

sampleSchemes *Demonstrate different sampling techniques*

Description

Using a Graphical User Interface (GUI) this function demonstrates the differences between several different sampling methods

Usage

```
sampleSchemes(gui=TRUE, sampleType = 0, sampleSize = 30, clusterSize = 5, ...)
```

Arguments

gui	An indicator of whether the interface should be activated
sampleType	Type of sampling to perform: 0=Simple random sampling, 1=Stratified by dorm, 2=Stratified by student type, 3=Cluster sampling by floor, 4=Synthetic sample
sampleSize	Number of students to sample
clusterSize	Number of clusters to sample or step size for synthetic sample
...	Additional arguments...

Details

This function produces two windows. The control window is a Tk window in which the type and size of sample can be specified. For the simple and stratified samples, the sample size n is taken as input. For the cluster sample, the number of clusters is taken, and for the synthetic sample, the number taken is how many rooms to skip before taking the next room.

The second window is a plot of the rooms, with the red rooms being the ones sampled.

Note

For the stratified random samples, if the n value doesn't divide evenly between the strata, then the extra n -values are assigned randomly, so some strata may have 1 more/less than others.

Also, for the synthetic sample, the starting point is randomly chosen on the first floor of the first building. the buildings are processed top to bottom, left to right, but within the buildings they are sample starting at the left side of the first floor.

Author(s)

Mohamed Abdoell <mohamed.abdoell@dal.ca> and Sam Stewart <samstewart11@gmail.com>

References

Heckard, Robert F., and Utts, Jessica M. Mind on Statistics. Thompson Brooks/Cole. Canada: 2007.

`sumSquares`*A Graphical Demonstration of Least Squares Regression*

Description

This function demonstrates the idea of simple logistic regression graphically, and allows the user to manipulate the line of best fit (the regression line) and see its effect on the error sum of squares.

Usage

```
sumSquares(x, y)
```

Arguments

<code>x</code>	A vector containing the covariates for the regression.
<code>y</code>	A vector containing the responses for the regression.

Details

The function performs a simple linear regression in the form $y = b_0 + b_1x$. The result is three plots: the top left is a simple plot of the data points with the regression line fitted. The top right plot is the same plot, but with squares drawn to connect each point to the fit. These squares can be thought of as visualizations of the residuals squared, which is how the Residual Sum of Squares is calculated. The bottom box prints individual sums of squares for some of the datapoints, and at the bottom prints the total sum of squares.

The top left box is the "Active" box, in that it has the regression line that can be manipulated. Both the intercept and the slope can be changed by clicking on the line. If the line is clicked near the y-axis then the regression line will turn blue, indicating that the *intercept* is being changed. Clicking anywhere on the plot after the line turns blue will cause the intercept to change so that the new regression line will hit that point. Clickinh on the line away from the y-axis will cause the line to turn red, indicating that the *slope* is being changed. Again, click on the plot and the slope will change so that the new regression line will hit that point.

The point of this function is to notice that, regardless of how the line is changed, a smaller sum of squares cannot be attained, and to demonstrate the idea of residual sum of squares graphically.

Author(s)

Mohamed Abdoell <mohamed.abdoell@dal.ca> and Sam Stewart <samstewart11@gmail.com>

tau

Finding the non-centrality parameter

Description

An iterative, brute-force function to calculate the non-centrality parameter of the chi-squared distribution required to achieve the given power and significance level.

Usage

```
tau(k, alpha, beta)
```

Arguments

k	degrees of freedom
alpha	The significance level
beta	1-power

Details

This function is designed for the calculation of sample sizes in survival analysis. It calculates the non-centrality parameter of a chi-squared distribution, given the significance, power and degrees of freedom.

Value

A scalar value representing the non-centrality parameter is returned.

Author(s)

Mohamed Abdoell <mohamed.abdoell@dal.ca> and Sam Stewart <samstewart11@gmail.com>

References

Makuch, Robert W. and Simon, Richard M. "Sample Size Requirements for Comparing Time-To-Failure Among k Treatment Groups". Chron Dis. 35. pg 861-867/ 1982.

Index

*Topic **datasets**

agpop, 1
MFSV, 13

*Topic **data**

sumSquares, 27

*Topic **design**

BlandAltman, 3
minimAllocGUI, 13
ratioCompare, 19
rocCurve, 24
sampleAssign, 25

*Topic **distribution**

histSample, 10
multHypothesis, 15
myHist, 16
powerDemo, 18

*Topic **package**

GUIdemo, 9

*Topic **regression**

outlierTest, 17

*Topic **survey**

bootSequence, 5
bootSingle, 6
cltDemo, 8

*Topic **survival**

caseRatio, 7

*Topic **univar**

confInt, 8
menuSampleSize, 12
reactGame, 20
rhoRange, 21
rhoScale, 23
sampleSchemes, 26
tau, 28

agpop, 1, 13

BlandAltman, 3, 13
bootSequence, 5, 6
bootSingle, 5, 6

caseRatio, 7

cltDemo, 8

confInt, 8

GUIdemo, 9

histSample, 10

identify, 17

menuSampleSize, 12

MFSV, 2, 4, 13

minimAllocGUI, 13

multHypothesis, 15

myHist, 16

outlierTest, 17

powerDemo, 18

ratioCompare, 19

reactGame, 20

rhoRange, 11, 21, 24

rhoScale, 22, 23

rocCurve, 24

sampleAssign, 25

sampleSchemes, 26

sumSquares, 27

tau, 28