russmisc-2.15.1

December 24, 2012

`aspect` *Solve right-triangles systems involving (monitor) aspect ratios*

Description

I designed this to calculate values for monitor aspect ratios, but I imagine it is more general. Given any two or three values, solve for the missing values. Note, the values are not checked so the principle of GIGO applies.

Usage

```
aspect(w = NULL, h = NULL, d = NULL, aspect = NULL)
```

Arguments

`w`	Width
`h`	Height
`d`	Diagonal
`aspect`	Aspect ratio (w/h)

Value

numeric The length of the unspecified side

References

http://en.wikipedia.org/wiki/Pythagorean_theorem

Examples

```
pythag(a=3,c=pythag(3,4))
```

aspell_add_word *Add a word to the end of an aspell dictionary*

Description

Actually this adds the value in the 'word' parameter to the end of any file specified by the 'file' argument.

Usage

```
aspell_add_word(word, dict)
```

Arguments

word	The word to be added to the dictionary
dict	The location of the dictionary

Examples

```
#mydictloc <- "c:\\rsp\\tools\\aspell\\bin\\aspell.exe"
#aspell_add_word("pythagorean",mydictloc)
```

bmp.save *Save current plot as a bmp*

Description

Save current plot as a bmp

Usage

```
bmp.save(filename, ...)
```

Arguments

filename	The filename for the saved plot
...	options passed to dev.copy

center *Center a variable*

Description

By centering I mean take the numeric variable and subtract the mean from each value. Then return each value.

Usage

```
center(x)
```

Arguments

x	Variable to be centered

Value

Numeric x - mean(x)

Examples

```
center(c(1,2,3))
```

checkFarm *Check a farm*

Description

See farm() for details on farms. This function checks for a file based on the farmName parameter called farmName.res.Rda. If that file exists it loads it and returns the object stored by the farm in the object farmName. If that file does not exist, then the farm is not done processing, and a warning and NULL are returned. Note that a rapid loop through checkFarm() without Sys.sleep produced an error during development.

Usage

```
checkFarm(farmName)
```

Arguments

farmName	This is the name of the farm, used for creating and destroying filenames. This should be saved from when the farm() is created

See Also

farm waitForFarm

Examples

```
.tmp <- "This is a test of farm()"
 exampleFarm <- farm("
 print(.tmp)
helloFarm <- 10+2
farmName <- helloFarm
save(farmName,file=farmResult)
")
example.result <- checkFarm(exampleFarm)
while (is.null(example.result)) {
example.result <- checkFarm(exampleFarm)
Sys.sleep(1)
}
print(example.result)
```

Conv2Dto3D *Calculate a real world position of a projected object*

Description

Given the position of an object on the projection plane, the real depth of the object, and the depth of the projection plane, calculate the position of the real-world object

Usage

```
Conv2Dto3D(ScreenPosition, TargetDepth, ScreenDepth)
```

Arguments

ScreenPosition	The position of the object on the screen
TargetDepth	The targets' position in depth coordinates
ScreenDepth	The position of the projection plane in depth coordinates

---

Conv3Dto2D *Calculate the projection position of a real world object*

Description

Given the position of an object in 3D space and the depth of the projection plane, calculate the position of the projection of that object on projection plane

Usage

```
Conv3Dto2D(TargetPosition, TargetDepth, ScreenDepth)
```

Arguments

TargetPosition	The targets' position in width and/or height coordinates
TargetDepth	The targets' position in depth coordinates
ScreenDepth	The position of the projection plane in depth coordinates

Value

numeric Projection position in the coordinate type specified in TargetPosition

---

`convertID` *Convert Excel Column ID*

Description

This function converts R numeric column IDs to Excel letter IDs and vice versa.

Usage

```
convertID(ID)
```

Arguments

`ID`	Either a character or numeric value specifying the ID to convert.

Value

character or numeric; the reverse of what was specified by ID

Note

This function is not properly vectorized, it can only handle a single ID

`debackslash` *Debackslash*

Description

Take the contents of the clipboard and replace backslashes with forward slashes

Usage

```
debackslash()
```

degrees *Convert radians to degrees*

Description

Convert radians to degrees

Usage

```
degrees(radians)
```

Arguments

radians	A value in radians

Value

The value provided in radians converted to degrees

See Also

radians

Examples

```
degrees(pi)
```

dist3D.l *Helper function for dist3D to handle lists*

Description

Helper function for dist3D to handle lists

Usage

```
dist3D.l(point1, point2)
```

Arguments

point1	I think it wants a list. Frankly, I don't remember how this works.
point2	I think it wants a list. Frankly, I don't remember how this works.

Value

numeric ... I think

Note

TODO: Figure out how this works.

dist3D *Calculate the distance between two 3D points*

Description

Calculate the distance between two 3D points

Usage

```
dist3D(point1, point2)
```

Arguments

point1	The location of the first point as a numeric vector of length 3 c(X,Y,Z)
point2	The location of the second point as a numeric vector of length 3 c(X,Y,Z)

Value

numeric distance between two points

References

http://www.calculatorsoup.com/calculators/geometry-solids/distance-two-points.php

Examples

```
dist3D(c(5,6,2),c(-7,11,-13))
dist3D(list(c(0,0),c(0:0),c(0,0)),list(1:2,3:4,5:6))
dist3D(list(5,6,2),list(-7,11,-13))
```

droplead0 *droplead0*

Description

Drop a leading zero from a number and convert to a character string, as in reporting p-values.

Usage

```
droplead0(val)
```

Arguments

val	The numeric value from which to drop the lead 0

factory *Catch errors and warnings and store them for subsequent evaluation*

Description

Factory is not my code. Factory generates a function which is appropriately wrapped by error handlers. The result produced is a list. This is a nice way to recover from a loop problems that may have occured during loop evaluation. Check the references for additional related functions. I have not included the other factory functions because they did not play well with the return item as an S4 object.

Usage

```
factory(fun)
```

Arguments

fun	The function to be turned into a factory

Value

list[3]: item 1 is the result from the function and $warn and $err contain warning and error messages respectively.

Author(s)

Martin Morgan; Packaged by Russell S. Pierce

References

http://stackoverflow.com/questions/4948361/how-do-i-save-warnings-and-errors-as-output-from-a-f

Examples

```
f.log <- factory(log)
f.log("a")
f.as.numeric <- factory(as.numeric)
f.as.numeric(c("a","b",1))
```

farm *Create a farm*

Description

A farm is an external self-terminating instance of R to solve a time consuming problem in R. Think of it as a (very) poor-person's multi-core. For a usage example, see checkFarm. Known issues: May have a problem if the library gdata has been loaded.// If a farm produces warnings or errors you won't see them If a farm produces an error... it never will produce a result.

Usage

```
farm(commands,
  farmName = paste("farm-", as.integer(Sys.time()) + runif(1), sep = ""),
  Rloc = NULL)
```

Arguments

commands	A text string of commands including line breaks to run. This must include the result being saved in the object farmName in the file farmResult (both are variables provided by farm() to the farm).
farmName	This is the name of the farm, used for creating and destroying filenames. One is randomly assigned that is plausibly unique.
Rloc	The location of R.exe. The default loads the version of R that is stored in the windows registry as being \"current\".

Value

The farm name is returned to be stored in an object and then used in checkFarm()

See Also

checkFarm waitForFarm

---

fft.binFreqs *Report the frequencies of the the bins in an FFT*

Description

Report the frequencies of the the bins in an FFT

Usage

```
fft.binFreqs(sfreq, Nsample)
```

Arguments

sfreq	numeric The number of samples per unit of time, e.g. Hz
Nsample	numeric The number of samples in the time series

Note

This code might not work

fft.binN *Find the number of FFT bins*

Description

Find the number of FFT bins

Usage

```
fft.binN(Nsample)
```

Arguments

Nsample	numeric The number of samples in the time series

fft.binSize *Find the size of an FFT bin*

Description

Find the size of an FFT bin

Usage

```
fft.binSize(sfreq, Nsample)
```

Arguments

sfreq	numeric The number of samples per unit of time, e.g. Hz
Nsample	numeric The number of samples in the time series

glibrary *Try to load a library, if that fails, install it, then load it.*

Description

glibrary short for (get)library. The primary aim of this function is to make loading packages more transparent. Given that we know we want to load a given package, actually fetching it is a formality. glibrary skims past this formality to install the requested package.

Usage

```
glibrary(..., lib.loc = NULL, quietly = FALSE,
  warn.conflicts = TRUE, pickmirror = FALSE,
  countrycode = "us")
```

Arguments

...	comma seperated package names
lib.loc	See require
quietly	See require
warn.conflicts	See require
pickmirror	If TRUE, glibrary allows the user to select the mirror, otherwise it auto-selects on the basis of the country code
countrycode	glibrary compares this value to results from getCRANmirrors() to select a mirror in the specified country

Value

logical; TRUE if glibrary was a success, an error if a package failed to load

Note

keep.source was an arguement to require that was deprecated in R 2.15

This warning `Warning in install.packages: InternetOpenUrl failed: 'The operation timed out'` indicates that the randomly selected repository is not available. Check your internet connection. If your internet connection is fine, set pickmirror=TRUE and manually select an operational mirror.

Examples

```
#glibrary(lattice,MASS) #not run to prevent needless dependency
```

---

in2m *Convert inches to meters*

Description

Convert inches to meters

Usage

```
in2m(inches)
```

Arguments

inches	A value in inches

Value

numeric a value in meters

km2mi *Convert kilometers to miles*

Description

Convert kilometers to miles

Usage

```
km2mi(km)
```

Arguments

km	numeric value in kilometers (km)

Value

numeric a value in miles

latex.p *Format a p-value for LaTex*

Description

All ps greater than .01 will be reported exactly to two decimal places. All ps greater than .001 but less than .01 will be reported exactly to 3 decimal places. All other ps will be reported at p < .001.

Usage

```
latex.p(p.val)
```

Arguments

p.val	A numeric p value

latexFigure *Insert a figure in LaTex*

Description

Insert a figure in LaTex

Usage

```
latexFigure(file, caption = NULL,
  placementSpecifier = "tdp", clearpage = FALSE)
```

Arguments

file	The name of the picture to insert
placementSpecifier	Options for LaTex that go in brackets when starting a figure. Tells LaTex where to put the picture. Note that the placement specifier here is the latex default, tdp. When using many images other packages, e.g. floats, morefloats may be useful. Alternatively, a \clearpage can clear out the floats.
clearpage	Whether to trigger a \clearpage after the figure.
caption	(optional) The caption for the figure in LaTex

---

latinsquare.digram *Create a digram-balanced latin square*

Description

This function creates a digram-balanced latin square as described in Keppel & Wickens, 2004 pg 384-386. The condition orders for subjects are listed by rows, such that row one is the order of conditions for subject 1, and so on. This method of counter-balancing is preferred to so-called cyclic counter-balancing because it balances the order of presentation such that conditions occur in each position equally and they also balance the order of presentation such that each condition occurs before and after each other presentation an equal number of times. See Keppel & Wickens, 2004 for details.

Usage

```
latinsquare.digram(conds)
```

Arguments

conds	A vector of condition IDs or condition names

Details

This is the deprecated name for `latinSquareDigram`

Value

Digram balanced latin squares come in pairs when there are an odd number of conditions, but only one square when there are an even number of conditions. This function returns a list with a square (in a matrix) for each item.

latinSquareDigram *Create a digram-balanced latin square*

Description

This function creates a digram-balanced latin square as described in Keppel & Wickens, 2004 pg 384-386. The condition orders for subjects are listed by rows, such that row one is the order of conditions for subject 1, and so on. This method of counter-balancing is preferred to so-called cyclic counter-balancing because it balances the order of presentation such that conditions occur in each position equally and they also balance the order of presentation such that each condition occurs before and after each other presentation an equal number of times. See Keppel & Wickens, 2004 for details.

Usage

```
latinSquareDigram(conds)
```

Arguments

conds	A vector of condition IDs or condition names

Value

Digram balanced latin squares come in pairs when there are an odd number of conditions, but only one square when there are an even number of conditions. This function returns a list with a square (in a matrix) for each item.

Author(s)

Russell S. Pierce <Russell.S.Pierce@gmail.com>

References

Keppel & Wickens, 2004 pp 384-386

Examples

```
latinsquare.digram(1:3)
latinsquare.digram(LETTERS[1:4])
```

lawOfCos *Use the law of sines Use the law of cosines to calculate an angle*

Description

Given that A, B, and C are the legs of a triangle solve the the opposite angles

Usage

```
lawOfCos(A, B, C)
```

Arguments

A	numeric Length of side A
B	numeric Length of side B
C	numeric Length of side C

Value

list(a=numeric angle opposite A,b=numeric angle opposite B,c=numeric angle opposite C)

References

http://oakroadsystems.com/twt/solving.htm#eq30

Examples

```
lawOfCos(3,4,5)
lawOfCos(3,3,3)
stimsize.val <- 10 #the stimulus size
dist <- 15 #the distance
visangle(10,15)-lawOfCos(stimsize.val,pythag(stimsize.val/2,dist),pythag(stimsize.val/2,dist))$a
offset <- 5
visangle(10,15,5)-lawOfCos(stimsize.val,pythag(stimsize.val/2+offset,dist),pythag(stimsize.val/2-offset,dis
```

---

longframe *Transform a wide dataset into a long dataset.*

Description

There is almost certainly a better way to do this. The aim here is to turn a wide dataset into a long dataset.

Usage

```
longframe(wide.data, btwnsubnames, wide.var.names,
  value.name = NA, dropcol = c())
```

Arguments

wide.data	The wide dataset
btwnsubnames	The between subject identifiers
wide.var.names	The variables that are in the wide format
value.name	unknown
dropcol	Columns to drop from the long format

See Also

widenames

Examples

```
widenames(c("Age","Workload"),c(2,2))
```

m2in *Convert meters to inches*

Description

Convert meters to inches

Usage

```
m2in(m)
```

Arguments

m	A value in meters

Value

numeric a value in inchs

MAD *Calculate the median absolute deviation (MAD)*

Description

Calculate the median absolute deviation (MAD)

Usage

```
MAD(x)
```

Arguments

x	A set of numeric scores

makewave *Make wave*

Description

Generate a vector of numeric data points representing a sine wave

Usage

```
makewave(freq, phase, amp, Nsamples = time * samplerate,
  samplerate, time = Nsamples/samplerate,
  as.time.series = TRUE)
```

Arguments

freq	The frequency of cycles per unit time (e.g. Hz) of the sine wave
phase	The frequency in degrees of the sine wave
amp	The amplitude of the sine wave
Nsamples	The number of samples to generate, this defaults to the time * sample rate.
samplerate	The number of samples to create per unit of time
time	The number of units of time represented by the generated sine wave
as.time.series	Should the result be returned as a time series ts()

Value

A numeric vector or a time series representing the specified sine wave

Note

coh is calculated as coh[, i + (j - 1) * (j - 2)/2] <- Mod(pgram[, i, j])^2/(spec[, i] * spec[, j]) Mod(pgram[, i, j])^2 #cross-amplitude values

Examples

```
plot(makewave(2,0,3.889*1.6,180,60))
sp <- spectrum(ts.union(makewave(15,30,1,180,60),makewave(15.01,60,1,180,60)),span=c(3),taper=0)
with(sp,data.frame(freq=freq,sta.freq=freq/60,phase=round(phase,4),coh=round(coh,4),phase.deg=round(phase/
plot(sp$freq,sp$phase/(2*pi)*180,type="l")
plot(sp,plot.type="phase")
plot(sp,plot.type="coh")
plot(ts.union(makewave(15,30,1,180,60),makewave(15.01,60,1,180,60)))
Mod(mvfft(ts.union(makewave(15,30,1,180,60),makewave(15.01,60,1,180,60))))
```

MEANS.STD.CORR *Write out the variables from X in a MPLUS MEANS STD CORR format*

Description

Take data.frame x which only has the variables of interest and convert to MEAN STD CORR format. Note this has not been tested at all. Also note that this will overwrite any pre-existing file.

Usage

```
MEANS.STD.CORR(x, filename = "output.dat")
```

Arguments

x	The data.frame with the variables to export
filename	The filename to export to.

Examples

```
x <- data.frame(X1=rnorm(20),X2=rnorm(20),X3=rnorm(20)) #example data
MEANS.STD.CORR(x)
```

mi2km *Convert miles to kilometers*

Description

Convert miles to kilometers

Usage

```
mi2km(mi)
```

Arguments

mi	numeric value in miles (mi)

Value

numeric a value in km

MSE *Calculate the mean squared error*

Description

Calculate the mean squared error

Usage

```
MSE(x, y, ...)
```

Arguments

x	Numeric data series
y	Numeric data series
...	further arguments for mean

Value

numeric

popscale *Population scale/Z scores*

Description

The built-in R function scale uses the sample standard deviation when its scale option is set to TRUE. This function uses the population standard deviation instead.

Usage

```
popscale(x, center = TRUE, scale = TRUE)
```

Arguments

x	A numeric vector, matrix, or data.frame
center	A logical indicating whether the scores in the columns in x should have their column means subtracted
scale	A logical indicating whether the scores in the columns in x should be divided by their column standard deviations

Author(s)

Borrowed from Ryne Sherman on 2012/09/06, his function is named scale2

popsd *Calculate the population standard deviation of x Usage popsd(x)*

Description

Calculate the population standard deviation of x Usage popsd(x)

Usage

```
popsd(x, nomiss = 0.8)
```

Arguments

x	A numeric vector
nomiss	A numeric vector specifying the proporiton of valid cases in x (i.e. data that must not be NA) for the sd to be returned

Note

R's built-in sd function divides the sum of the squared deviations from the mean by the number of observations minus 1 (N-1). However, there are times where one would prefer to use the formula with N in the denominator (e.g. if one is working with the entire population of scores). This function does just that.

Author(s)

Borrowed from Ryne Sherman on 2012/09/06

product *Take the product of all values in a vector*

Description

Take the product of all values in a vector

Usage

```
product(x)
```

Arguments

x	A vector of type numeric

Value

numeric The result of systematically multiplying the items in the vector x.

Examples

```
product(1:3)
product(c(2,2,4))
```

pythag *Calculate the pythagorean theorem to solve for A, B, or C*

Description

Given that A and B are the legs of a right triangle and C is its hypotenuse: solve for A, B, or C given any other two.

Usage

```
pythag(a = NULL, b = NULL, c = NULL)
```

Arguments

a	Leg 1
b	Leg 2
c	Hypotenuse

Value

numeric The length of the unspecified side

References

http://en.wikipedia.org/wiki/Pythagorean_theorem

Examples

```
pythag(a=3,c=pythag(3,4))
```

`r2t` *Convert r to a t*

Description

Convert r to a t

Usage

```
r2t(r, df)
```

Arguments

`r`	The correlation coefficient value
`df`	the df for the correlation coefficient

`radians` *Convert degrees to radians*

Description

Convert degrees to radians

Usage

```
radians(degrees)
```

Arguments

`degrees`	A value in degrees

Value

The value provided in degrees converted to radians

Examples

```
radians(90)
```

russmisc-package *russmisc-package placeholder*

Description

A repository for a variety of useful functions.

Details

A respository for a variety of useful functions which may be of interest or otherwise reused. Unless otherwise stated in the function description, all functions are authored by Russell S. Pierce.

Author(s)

Russell S. Pierce <Russell.S.Pierce@gmail.com>

russmisc.refresh *Refresh the russmisc package*

Description

This is just a convenience to unload an reload the russmisc package, useful for rapid development

Usage

```
russmisc.refresh()
```

simpleFarm *Create a one-line simple farm*

Description

This is a convience wrapper function that uses farm to create a single farm appropriate for processing single line commands.

Usage

```
simpleFarm(command,
  farmName = paste("farm-", as.integer(Sys.time()) + runif(1), sep = ""),
  Rloc = NULL)
```

Arguments

command	A single command
farmName	This is the name of the farm, used for creating and destroying filenames. One is randomly assigned that is plausibly unique.
Rloc	The location of R.exe. The default loads the version of R that is stored in the windows registry as being \"current\".

Value

The farm name is returned to be stored in an object and then used in checkFarm()

See Also

farm, checkFarm, and waitForFarm

Examples

```
a <- 5
b <- 10
farmID <- simpleFarm("a + b")
waitForFarm(farmID)
```

SSE *Calculate the sum of squared errors*

Description

Calculate the sum of squared errors

Usage

```
SSE(x, y, ...)
```

Arguments

x	Numeric data series
y	Numeric data series
...	further arguments for mean

Value

numeric

---

stimsize *Calculate a stimulus size given a visual angle and viewing distance*

Description

Calculate a stimulus size given a visual angle and viewing distance

Usage

```
stimsize(visangle.val, viewdist.val)
```

Arguments

visangle.val	A value in degrees of visual angle
viewdist.val	A viewing distance

Value

The stimulus size that when centrally presented would have a visual angle (in degrees) of visangle.val. Stimulus size is reported in the same units as viewdist.val.

See Also

viewdist stimsize

Examples

```
stimsize(visangle(10,100),100)
```

---

t2r *Convert t to a r*

Description

Convert t to a r

Usage

```
t2r(t, df)
```

Arguments

t	The t statistic
df	The df for the t statistic

---

table.png *Print LaTex Table Direction to an Image*

Description

Print LaTex Table Direction to an Image

Usage

```
table.png(obj, name)
```

Arguments

obj	The matrix to turn into a table
name	The name of the png to produce (without extention)

Author(s)

Found here: http://stackoverflow.com/questions/9298765/print-latex-table-directly-to-an-image-png-or-other

trimval *Trim values Usage trimval(x, tr=.2, na.rm=TRUE)*

Description

Trim values Usage trimval(x, tr=.2, na.rm=TRUE)

Usage

```
trimval(x, tr = 0.2, na.rm = TRUE)
```

Arguments

x	A numeric vector to be trimmed
tr	The proportion of values in x to be trimmed
na.rm	A logical indicating whether missing values should be removed prior to trimming

Author(s)

borrowed from Ryne Sherman on 2012/09/06. Modified to return NA for removed values by Russell S. Pierce

unfactor *Unfactor a vector*

Description

Take a factor vector and return a value a vector of either type character or numeric by replacing the factors with their associated labels. If all label names are numeric, a numeric vector will be returned, otherwise a character vector will be returned.

Usage

```
unfactor(factors)
```

Arguments

factors	The vector of factors to be unfactored

Value

character or numeric

Examples

```
unfactor(factor(c(3,2,1)))
```

uniquePerms *Get a set of minimially replicated permutations*

Description

This function provides a random set of sequences from the full possible set of permuations. If you request more sequences than there are unique ones it will give you a full set of unique permutations/sequences (or more than one) and then randomly select the remaning sequences. The arguments are conds (like in latinsquare.digram) and N (number of sequences requested)

Usage

```
uniquePerms(conds, N)
```

Arguments

conds	(vector) A of condition IDs or condition names
N	(numeric) The number of sequences desired

Value

N x conds (matrix) of sequences (by rows)

Author(s)

Russell S. Pierce <Russell.S.Pierce@gmail.com>

Examples

```
uniquePerms(LETTERS[1:4],12)
uniquePerms(LETTERS[1:4],36)
```

updateTex *Update a .tex file from .rnw file if needed*

Description

Check the file dates. If .Rnw is newer than .Tex, then re-weave.

Usage

```
updateTex(filename)
```

Arguments

filename	character the root of the filename, e.g. "writing" when refering to "Writing.Rnw"

viewdist — *Calculate a viewing distance given a visual angle and stimulus size*

Description

Calculate a viewing distance given a visual angle and stimulus size

Usage

```
viewdist(visangle.val, stimsize.val)
```

Arguments

visangle.val	A value in degrees of visual angle
stimsize.val	A stimulus size

Value

The viewing distance of an object of stimsize.val that, when centrally presented, would have a visual angle (in degrees) of visangle.val. Viewing distance is reported in the same units as stimsize.val.

Author(s)

Russell S. Pierce <Russell.S.Pierce@gmail.com>

See Also

viewdist visangle

Examples

```
viewdist(visangle(10,100),10)
```

visangle — *Calculate visual angle given a stimulus size and viewing distance*

Description

Calculate visual angle given a stimulus size and viewing distance

Usage

```
visangle(stimsize.val, viewdist.val, offset = 0)
```

Arguments

stimsize.val	The stimulus size
viewdist.val	The viewing distance
offset	The offset of the center of the object from the center of the viewing angle

Value

Degrees in visual angle for the given stimulus size

Note

The units of measure should be the same for both parameters

See Also

stimsize viewdist link{visangle3D}

Examples

```
visangle(10,100)
visangle(visangle(0.75438,45),viewdist(visangle(0.75438,45),0.75438))
```

---

visangle3D.l *Helper function for visangle3D to handle lists*

Description

Helper function for visangle3D to handle lists

Usage

```
visangle3D.l(viewerloc, stimloc, stimdim)
```

Arguments

viewerloc	The location of the center of the viewer as a list
stimloc	The location of the center of the stimulus as a list
stimdim	The stimulus dimensions as a numeric list

---

visangle3D *Calculate visual angle given a 3D viewer position, stimulus size, and stimulus position*

Description

This function calculates the visual angle of a stimulus given a viewer position, stimulus size, and stimulus position. Here I make fewer assumptions and calculate the distance between the viewer and the edges of the object in order to use the law of cosines to calculate the associated angle.

Usage

```
visangle3D(viewerloc, stimloc, stimdim)
```

Arguments

viewerloc	The location of the center of the viewer as a numeric vector of length 3 c(X,Y,Z)
stimloc	The location of the center of the stimulus as a numeric vector of length 3 c(X,Y,Z)
stimdim	The stimulus dimensions as a numeric vector of length 2 c(width,height)

Value

Degrees in visual angle for the given stimulus dimentions and positions of viewer and stimuli

Note

The units of measure should be the same for all parameters. Z is assumed to be the depth plane.

See Also

visangle

Examples

```
#corresponds to an offset of 10 and 10 and a viewer distance of 20
viewerloc <- c(0,0,0)
stimloc<-c(10,10,-20)
stimsize.val <- 10
stimdim <- c(stimsize.val,1)
visangle(stimsize.val,20,pythag(stimloc[1],stimloc[2]))
visangle3D(viewerloc,stimloc,stimdim)
#corresponds to an offset of 10 and 0 and a viewer distance of 20
viewerloc <- c(0,0,0)
stimloc<-c(10,0,-20)
stimsize.val <- 10
stimdim <- c(stimsize.val,1)
visangle(stimsize.val,20,pythag(stimloc[1],stimloc[2]))
visangle3D(viewerloc,stimloc,stimdim)
visangle3D(list(c(1,1),c(1,1),c(1,1)),list(c(2,2),c(2,2),c(2,2)),list(c(3,3),c(3,3)))
visangle3D(list(1,2,3),list(3,4,5),list(2,2))
```

---

waitForFarm *Wait for a farm result*

Description

This function repeatedly checks for a farm, when the farm is found it returns the harvest (the farm result object). If the farm terminated with an error or there is some other sort of coding error, waitForFarm will be an infinate loop. As checkFarm produces errors on checks when the harvest is not ready, waitForFarm hides these errors in the factory error-catching wrapper.

Usage

```
waitForFarm(farmName, noCheck = FALSE)
```

Arguments

farmName	This is the name of the farm, used for creating and destroying filenames. This should be saved from when the farm() is created
noCheck	If this value is TRUE the check for the farm's .r is skipped. If it is FALSE, the existance of the appropriate .r is checked for before entering a potentially unending while loop.

---

widenames *Generate Names for a Wide Stuctured Dataset*

Description

Generate Names for a Wide Stuctured Dataset

Usage

```
widenames(names, levels)
```

Arguments

names	A vector of type character with the names of the factors
levels	The number of levels for each factor listed in names

See Also

longframe

Examples

```
widenames(c("Age","Workload"),c(2,2))
```

---

winvar *Calculate the winsorized variance of x Usage winvar(x, tr=.2, na.rm=TRUE) This function is borrowed directly from Rand Wilcox's function and is kept with this set of functions for capability with a large number of functions.*

Description

Calculate the winsorized variance of x Usage winvar(x, tr=.2, na.rm=TRUE) This function is borrowed directly from Rand Wilcox's function and is kept with this set of functions for capability with a large number of functions.

Usage

```
winvar(x, tr = 0.2, na.rm = TRUE)
```

Arguments

x	A numeric vector of which to get the winsorized variance
tr	The proportion of scores to winsorize
na.rm	A logical indicating whether missing values should be removed prior to calculation.

Author(s)

Rand Wilcox, borrowed from Ryne Sherman on 2012/09/06

Zmodified *Calculate the Iglewicz and Hoaglin (1993) Z modified*

Description

Calculate the Iglewicz and Hoaglin (1993) Z modified

Usage

```
Zmodified(x)
```

Arguments

x	A set of numeric scores

Index