

# Package ‘jvamisc’

September 30, 2015

**Version** 0.0.0.9007

**Date** 2015-09-30

**Title** A Collection of Utility Functions

**Author** Jean V. Adams [aut, cre]

**Maintainer** Jean V. Adams <jvadams@usgs.gov>

**Depends** R (>= 3.2.2)

**Imports** devtools, lubridate, maptools, MASS, plotrix, RColorBrewer,  
RCurl, rpart, seriation, sp, survey, twitterR

**Description** A collection of miscellaneous utility functions.

**LazyData** TRUE

**License** GPL

**URL** <https://github.com/JVAdams/jvamisc>

**NeedsCompilation** no

## R topics documented:

addedvar . . . . .	3
addhist . . . . .	3
AICc . . . . .	4
allcombs . . . . .	5
bcpCI . . . . .	6
binomCI . . . . .	7
blindcolz . . . . .	7
brewcol . . . . .	8
calcr2 . . . . .	8
capwords . . . . .	9
char2num . . . . .	10
cheat . . . . .	11
chi . . . . .	15
CI . . . . .	16
circles . . . . .	17
cleanup . . . . .	18
colr . . . . .	18
coordflip . . . . .	19
coordmove . . . . .	19
coordplot . . . . .	20

coordtri . . . . .	21
coordturn . . . . .	22
counties . . . . .	23
dfclip . . . . .	23
doy . . . . .	24
drawcell . . . . .	24
fill . . . . .	25
first . . . . .	26
formatdf . . . . .	26
getpackages . . . . .	27
inrange . . . . .	28
jvaFirst . . . . .	28
jvaLast . . . . .	29
jvamic . . . . .	29
jvamicenv . . . . .	29
jvanames . . . . .	30
jvatree . . . . .	30
Lakeabbs . . . . .	31
Lakenames . . . . .	31
last . . . . .	31
latlong2 . . . . .	32
lls . . . . .	33
map5 . . . . .	33
mapL . . . . .	34
mapSMR . . . . .	34
matrixtrim . . . . .	35
modecontin . . . . .	35
mrc . . . . .	36
mytable . . . . .	37
numbers2words . . . . .	37
pkgin . . . . .	38
pkgman . . . . .	38
pkgup . . . . .	39
plotblank . . . . .	39
plotcor . . . . .	40
plotdf . . . . .	41
predAntilog . . . . .	42
prettylog . . . . .	42
prettytable . . . . .	43
ratest . . . . .	44
recode . . . . .	45
segreg . . . . .	45
shadepoly . . . . .	46
showmarks . . . . .	47
states . . . . .	47
stringin . . . . .	48
trimspace . . . . .	49
tweethead . . . . .	49

addedvar

*Added Variable Plots of Predictors***Description**

Produces an added variable plot given 1 response and 2 or more predictors.

**Usage**

```
addedvar(Y, X, main = "")
```

**Arguments**

Y	A vector representing a single response.
X	Two or more predictor columns in a matrix or data frame.
main	Subtitle for the plot.

**Value**

A plot is sent to the current graphics device (no value is returned).

**Examples**

```
addedvar(Y=mtcars$hp, X=mtcars[, c("mpg", "displacement", "wt")],
main="Predicting horsepower from MPG, displacement, and weight")
```

addhist

*Add a Marginal Histogram***Description**

Add a marginal histogram to a plot.

**Usage**

```
addhist(x, y = NULL, type = "xy", nclass = 20, newmar = 0:1,
adj.fac = 1.05, xlab = "Frequency", ylab = "", fill = "gray")
```

**Arguments**

x	A numeric vector, the first set of data to be binned into a histogram.
y	A numeric vector, the second set of data to be binned into a histogram, default NULL.
type	A character scalar, indicating to which plot axes histograms should be added; "x" adds a histogram along the x-axis, "y" adds a histogram along the y-axis, "xy" (the default) adds a histogram along both axes.
nclass	An integer scalar, the target number of bins for the histogram, default 20.
newmar	A numeric vector of length 2, indicating new margins to use, default c(0, 1).

<code>adj.fac</code>	A numeric scalar, adjustment factor for extent of y-axis, default 1.05 (to ensure tallest bars in histogram are below axis box).
<code>xlab</code>	A character scalar, label for x-axis, default "Frequency".
<code>ylab</code>	A character scalar, label for y-axis, default "".
<code>fill</code>	A character or numeric scalar, color for filling in histogram bars, default "gray".

### See Also

[hist.](#)

### Examples

```
# fake data
xx <- rnorm(30)
yy <- runif(30)

# type "x"
layout(matrix(2:1, ncol=1), heights=c(1/5, 4/5))
par(mar=c(4, 4, 0, 1), cex=1, las=1)
plot(xx, yy)
addhist(xx, type="x")

# type "y"
layout(matrix(1:2, ncol=2), widths=c(4/5, 1/5))
par(mar=c(4, 4, 1, 0), cex=1, las=1)
plot(xx, yy)
addhist(yy, type="y")

# type "xy"
layout(matrix(c(2, 1, 0, 3), ncol=2),
  heights=c(1/5, 4/5), widths=c(4/5, 1/5))
par(mar=c(4, 4, 0, 0), cex=1, las=1)
plot(xx, yy)
usr <- par("usr")
mar <- par("mar")
addhist(xx, yy, type="xy")
```

---

AICc

---

*Comparison of Models using Akaike Information Criterion*


---

### Description

Compares a collection of statistical models using AIC.

### Usage

```
AICc(fitlist, corr = TRUE)
```

### Arguments

<code>fitlist</code>	A list of model fits to compare, e.g., <code>lm</code> , <code>glm</code> , <code>aov</code> objects.
<code>corr</code>	A logical indicating whether the AIC should be corrected for small sample size, default <code>TRUE</code> .

**Value**

Data frame with a row for each model being compared, ordered by either the uncorrected AIC (corr=FALSE) or the AIC corrected for small sample size (corr=TRUE). Columns include the number of observations (n), the number of parameters (p), the root mean squared error (rmse), the uncorrected AIC (aic), the AIC corrected for small sample size (aicc), the delta AIC (daic or daicc), and the weights of evidence (aicw or aiccw).

**Examples**

```
fit1 <- lm(hp ~ mpg + disp + wt, data=mtcars)
fit2 <- lm(hp ~ mpg + disp, data=mtcars)
fit3 <- lm(hp ~ mpg + wt, data=mtcars)
fit4 <- lm(hp ~ disp + wt, data=mtcars)
AICc(list(fit1, fit2, fit3, fit4))
```

---

allcombs	<i>All Combinations</i>
----------	-------------------------

---

**Description**

All possible combinations of the given number of items.

**Usage**

```
allcombs(num, from = 0, to = num)
```

**Arguments**

num	A scalar, the number of items.
from	A scalar, the minimum number of items in each combination, default 0.
to	A scalar, the maximum number of items in each combination, default num.

**Value**

A matrix with rows corresponding to each possible combination and columns corresponding to item number.

**References**

Based on a method posted by Petr Savicky on 20 Jul 2012 to R-help [\[link\]](#).

**Examples**

```
allcombs(3)
```

bcpCI

*Bias-Corrected Percentile Confidence Interval***Description**

Calculates the bias-corrected percentile confidence interval from a bootstrap sample.

**Usage**

```
bcpCI(tboot, orig, alpha = 0.05)
```

**Arguments**

tboot	A numeric vector of bootstrap estimates, typically \$t from the output of the <code>boot</code> function.
orig	A numeric scalar, the original estimate from the data, typically \$t0 from the output of the <code>boot</code> function.
alpha	A numeric scalar, the desired significance level for 100*(1-alpha)% confidence limits, default 0.05.

**Value**

A named numeric vector of length 2, with the lower and upper confidence limits.

**References**

Manly, Bryan F. J. 1997. Randomization, Bootstrap and Monte Carlo Methods in Biology. Chapman & Hall, London.

**Examples**

```
bcpCI(exp(rnorm(20)), 1)

## Not run:
# Bootstrap of the ratio of means using the city data
library(boot)
ratio <- function(d, w) sum(d$x * w)/sum(d$u * w)
results <- boot(city, ratio, R=999, stype="w")
results$t0
bcpCI(results$t, results$t0, 0.1)

## End(Not run)
```

---

binomCI	<i>Binomial Confidence Interval</i>
---------	-------------------------------------

---

**Description**

Calculates the binomial confidence interval from a sample, using the normal approximation. Uses Louis (1981) if only failures or only successes were observed.

**Usage**

```
binomCI(x, y = NULL, na.rm = TRUE, alpha = 0.05, prob = TRUE)
```

**Arguments**

x	A vector of 0s and 1s, or (if y is also given) a scalar, the number of successes.
y	A scalar, the number of failures, default NULL.
na.rm	A logical, whether to remove NAs from the data, default TRUE.
alpha	A scalar, the desired confidence level, default 0.05.
prob	A logical, whether to output results as probabilities, default TRUE, or counts.

**Value**

A named vector with the Mean, lower and upper confidence limits (L and U), and the number of observations N.

**References**

Thomas A. Louis. 1981. Confidence intervals for a binomial parameter after observing no successes. The American Statistician 35(3):154. <http://amstat.tandfonline.com/doi/abs/10.1080/00031305.1981.10479337?journal=ast>

**Examples**

```
binomCI(c(0, 0, 0, 0, 1, 1))
binomCI(2, 4, prob=FALSE)
```

---

blindcolz	<i>Color-blind Friendly Colors</i>
-----------	------------------------------------

---

**Description**

A vector with nine color-blind friendly colors in hex code.

**Format**

A character vector, length 9.

**Author(s)**

Masataka Okabe and Kei Ito.

**Source**

How to make figures and presentations that are friendly to color blind people, 20 November 2002, [\[link\]](#).

---

brewcol

*Assign Colors to Collection of Values Using Color Brewer*


---

**Description**

Assign a specified number of Color Brewer colors to a collection of values.

**Usage**

```
brewcol(x, n = 9, name = c("YlGnBu", "Oranges", "Greens", "Blues", "Greys",
"Dark2")[1])
```

**Arguments**

x	A numeric vector of values to which colors will be assigned.
n	An integer scalar indicating the number of unique colors to be assigned, default 9. The range of possible values depends on the palette named, typically n should be from 3 to 9.
name	A character scalar indicating the palette name to be passed to <a href="#">brewer.pal</a> , default "YlGnBu".

**Value**

A character vector of colors expressed as hexadecimal values with a "#" prefix.

**Examples**

```
x <- 1:20
mycol <- brewcol(x, 4, "Dark2")
plot(x, x, col=mycol, pch=16, cex=3)
```

---

calcr2

*Coefficient of Determination*


---

**Description**

Calculate the unadjusted and adjusted coefficient of determination ( $R^2$ ).

**Usage**

```
calcr2(fitted, observed, nparam)
```



**Arguments**

fitted	A vector of fitted values.
observed	A vector of observed values.
nparam	A scalar, number of parameters in the fitted model.

**Value**

A vector of unadjusted and adjusted coefficients of determination.

**Examples**

```
fit1 <- lm(mpg ~ cyl + disp, data=mtcars)
calcr2(fit1$fitted, mtcars$mpg, 3)

fit2 <- lm(mpg ~ cyl + disp + wt, data=mtcars)
calcr2(fit2$fitted, mtcars$mpg, 4)
```

---

capwords

*Capitalize Words*


---

**Description**

Capitalize the first letter of every word.

**Usage**

```
capwords(s, strict = TRUE)
```

**Arguments**

s	A vector of strings.
strict	A logical indicating whether other letters should be converted to lower case, default TRUE.

**Value**

A vector the same length as s.

**See Also**

[casefold](#), from which the function was derived.

**Examples**

```
capwords(c("using AIC for model selection"))
capwords(c("using AIC", "for MODEL selection"), strict=FALSE)
```

char2num

*Convert Character to Numeric***Description**

Convert a character vector to a numeric vector, save any non-numeric values to a separate vector of comments.

**Usage**

```
char2num(x, varname = NULL, pmissing = c("NA", ".", "", " ", "-"),
        justindex = FALSE)
```

**Arguments**

x	A character vector to be converted to numeric.
varname	A character scalar to be used as an identifier in the generated comment vector. If NULL (default), the name of x will be used. If FALSE, no varname will be used in the comment.
pmissing	A character vector of all possible values to be interpreted as missings (and, therefore, not be commented upon).
justindex	A logical scalar indicating if the output should just be the index of non-numeric values, default FALSE.

**Value**

By default, a list of length two with the converted numeric vector and the comment vector. If justindex=TRUE, a character vector the same length as x, indicating the elements with non-numeric values.

**Examples**

```
a <- c(">1.5", "3", "NA", "missing", ".", "12")
char2num(a)
char2num(a, FALSE)
char2num(a, "pH")
char2num(a, pmissing=c("NA", "missing", "."))
char2num(a, justindex=TRUE)

df <- data.frame(
  x1=c(NA, 8.2, 8.2, 8.2, 8.6, 8.1, 8.2),
  x2=c(NA, "83", "*", "83", "79-80", "NA", "83"),
  x3=c(NA, NA, NA, "9.4", "?", ">10", "6.6"))
vars <- c("x2", "x3")
tonum <- lapply(vars, function(v) char2num(df[, v], varname=v))
df2 <- df
df2[, vars] <- data.frame(sapply(tonum, "[", 1))
df2$comment <- apply(do.call(cbind, sapply(tonum, "[", 2)), 1,
  paste, collapse="")
```

## Description

A non-functioning function, which allows me to create a cheat sheet collection of examples.

## Usage

```
cheat()
```

## Examples

```
## Not run:

# read in xls files
library(XLConnect)
wb <- loadWorkbook("c:/temp/junk.xlsx")
dat <- readWorksheet(wb, sheet=getSheets(wb)[1], startRow=1)

### update package ###
pkgup("jvamic")
pkgin("jvamic")
del C:\Users\jvadams\*.gz
"C:\Program Files\R\R-3.2.2\bin\x64\R.exe" CMD build C:\JVA\GitHub\jvamic --resave-data
"C:\Program Files\R\R-3.2.2\bin\x64\R.exe" CMD check C:\Users\jvadams\jvamic_0.0.0.9007.tar.gz
pkgman("jvamic")

### Greek and math symbols ###
# http://www.decodeunicode.org/
plot(1, 1, xlab="Length (\U03BCm)", ylab="Temperature (\U00b0 C)",
main="Lambda squared=\U03BB\U00B2=\U03BB\U00B2")

### map scale and north arrow ###
library(GISTools)
map("state", region= "ohio")
maps::map.scale(x=-84, y=40, ratio=FALSE, relwidth=0.2)
north.arrow(xb=-83.4, yb=40.4, len=0.05, lab="N")

### error bars ###
x <- 1:10
y <- sample(10)
noise <- abs(rnorm(10))
plot(x, y, ylim=range(y-noise, y+noise))
arrows(x, lo, x, hi, length=0.1, angle=90, code=3)

### error shading ###
x <- 1:10
y <- sample(10)
noise <- abs(rnorm(10))
plot(x, y, ylim=range(y-noise, y+noise), type="n")
shadepoly(x, y, y-noise, y+noise)

### confidence limits ###
# of the mean
```

```

p <- predict(fit, interval="confidence")
# of a new observation
p <- predict(fit, interval="prediction")
# if you have a gam ...
p <- predict(fit, se.fit=TRUE)
pe <- p$fit
tt <- qt(1 - 0.05/2, fit$df.residual)
pl <- pe + tt*p$se.fit
pu <- pe - tt*p$se.fit

### multiple comparison Tukey test approach 1 ###
amod <- aov(breaks ~ tension, data=warpbreaks)
TukeyHSD(amod)

### multiple comparison Tukey test approach 2 ###
library(multcomp)
amod <- aov(breaks ~ tension, data=warpbreaks)
mc <- glht(amod, linfct=mcp(tension="Tukey"))
summary(amod)
summary(mc)
confint(mc)
dev.new()
plot(mc)

### set up two-way ANOVA with interactions ###
fit <- aov(y ~ f1 + f2 + f1:f2)
# set up linear hypotheses for all-pairs of both factors
wht <- glht(fit, linfct=mcp(f1="Tukey", f2="Tukey"))
# cf. Westfall et al. (1999, page 181)
summary(wht, test=adjusted("Shaffer"))

### label months on day of year or Julian day axis ###
plot(101:200, rnorm(100), axes=FALSE)
axis(1, at=doy(as.Date(paste(2000, 1:12, 1, sep="-")))-0.5,
labels=FALSE)
axis(1, at=doy(as.Date(paste(2000, 1:12, 15, sep="-"))),
labels=month.abb, tick=FALSE)
axis(2)
box()

### contour plot ###
library(akima)
y <- rnorm(50)
x <- runif(50)
z <- 2*x^2 - y^2 + 4
contour(interp(x, y, z, duplicate="mean"))

### get lat longs for locations ###
library(dismo)
geocode(c("1600 Pennsylvania Ave NW, Washington DC",
"Luca, Italy", "Kampala", "Antigo, WI"))

### get a lat long for a location ###
# from R-help post by Phil Spector, UC Berkeley, Mar 16, 2010
# https://stat.ethz.ch/pipermail/r-help/2010-March/232090.html
library(XML)
root <- xmlRoot(xmlTreeParse(

```

```

paste0("http://maps.google.com/maps/api/geocode/xml?address=",
"Antigo, WI", "&sensor=false"))
lat <- xmlValue(root[["result"]][["geometry"]][["location"]][["lat"]])
long <- xmlValue(root[["result"]][["geometry"]][["location"]][["lng"]])

### plot points on a map ###
library(RgoogleMaps)
MyMap <- GetMap.bbox(c(-80, -79), c(45, 46), maptype="terrain",
destfile="junk.png", zoom=8)
PlotOnStaticMap(MyMap, lat=seq(45, 46, 0.1), lon=seq(-80, -79, 0.1),
col="red", pch=16)

### make a quick map ###
library(ggmap)
qmap("Antigo, Wisconsin", zoom=14)

### convert between lat/long and projections ###
library(proj4)
project(xy, proj, inverse=FALSE, degrees=TRUE,
silent=FALSE, ellps.default="sphere")

### other map stuff of interest ###
# http://cartodb.com/
# https://github.com/Vizzuality/cartodb-r

### see the details of a function
methods(function)
package::function.default

### dendrogram reorder ###
x <- sample(100, 10)
names(x) <- x
hc <- hclust(dist(x))
dd <- as.dendrogram(hc)
dd.reorder <- reorder(dd, x, mean)
par(mfcol=1:2)
plot(dd, main="default dendrogram")
plot(dd.reorder, main="reordered")

### background jpeg image in plot ###
library(jpeg)
x <- rnorm(20)
y <- rnorm(20)
img <- readJPEG("C:/Users/Public/Pictures/Sample Pictures/Chrysanthemum.jpg")
plot(x, y, type="n")
pusr <- par("usr")
rasterImage(img, pusr[1], pusr[3], pusr[2], pusr[4])
points(x, y, pch=16, cex=3)

### fit all subsets models ###
# select all possible combinations of 8 independent variables
var.names <- names(train)[1:8]
comb <- as.data.frame(all.combs(8))
dimnames(comb)[[2]] <- var.names
fits <- vector("list", dim(comb)[1])
fits[[1]] <- lm(lpsa ~ 1, dat=train)
for(i in 2:length(fits)) {

```

```

fits[[i]] <- lm(formula=paste("lpsa ~",
paste(var.names[comb[i, ]==1], collapse=" + ")), dat=train)
}
comb2 <- comb
comb2$nx <- apply(comb, 1, sum)
# AIC
aic <- AICc(fits)
aic <- aic[order(as.numeric(row.names(aic))), -1]
comb2 <- data.frame(comb2, aic)
comb2 <- comb2[order(comb2$aicc), ]
comb2[comb2$aicc <= 2, ]
fits[[as.numeric(row.names(comb2)[1])]]

### regular expression examples ###
# get rid of spaces before commas or periods
t2 <- gsub("[[:space:]]\\. ", "\\. ", charvec)
gsub("[[:space:]]\\. ", "\\. ", t2)
# insert a space between all punctuation and letters
gsub("([[:punct:]])([[:alpha:]])", "\\1 \\2", charvec)
# add a period to any single alpha character
t2 <- gsub("([[:space:]])([[:alpha:]])", "\\1\\.\\2", charvec)
gsub("([[:space:]]).$", "\\1\\. ", t2)
# insert a space before and after an equal sign
t2 <- gsub("([[:alpha:]])([[:punct:]])=", "\\1 =", charvec)
gsub("=([[:alpha:]])([[:punct:]])", "= \\1", t2)
# make sure Jr has a period after it
gsub("Jr$", "Jr. ", t2)
# remove all apostrophes (and any surrounding spaces)
t2 <- gsub('[[:space:]]\\\'[[:space:]]', "", charvec)
t2 <- gsub('[[:space:]]\\\'', "", t2)
gsub('\\\'[[:space:]]', "", t2)
# cut off equal sign and everything after
gsub("=.+", "", charvec)
# replace all punctuation marks with spaces
gsub("[[:punct:]]", " ", charvec)
# get rid of leading and trailing white space
gsub("^\\s+|\\s+$", "", charvec)
# change double spaces to single spaces
gsub("[\\s]{2}", " ", charvec)

### convert a data frame to json ###
library(RJSONIO)
data <- toJSON(y)
cat(data, file="data.json")

### read in internet table ###
library(XML)
allTables <- readHTMLTable(
"http://en.wikipedia.org/wiki/United_States_presidential_election,_2012")
# Look at the allTables object to find the specific table we want
str(allTables)
# if you have problems reading the URL, you could try this ...
mylines <- readLines(url(
"http://en.wikipedia.org/wiki/United_States_presidential_election,_2012"))
closeAllConnections()
mylist <- readHTMLTable(mylines, asText=TRUE)
mytable <- mylist1$table

```

```

### allow users to browse to a file ###
library(tcltk)
myfile <- tk_choose.files(default="C:/JVA/*.csv")

### one slider ###
library(rpanel)
density.draw <- function(panel) {
  plot(density(panel$x, bw=panel$h))
  panel
}
panel <- rp.control(x=rnorm(50))
rp.slider(panel, h, 0.5, 5, log=TRUE, action=density.draw)

### two sliders ###
library(rpanel)
loess.draw <- function(panel) {
  plot(panel$x, panel$y)
  lines(loess.smooth(panel$x, panel$y, span=panel$s, degree=panel$d))
  panel
}
panel <- rp.control(x=rnorm(50), y=rnorm(50), s=rep(3, 50), d=1)
rp.slider(panel, s, 0.1, 10, showvalue=TRUE, action=loess.draw)
rp.slider(panel, d, 1, 2, showvalue=TRUE, action=loess.draw)

### animation ###
for(i in 1:10) {
  dev.new()
  plot(1:10, 1:10, type="l")
  points(i, i, pch=16, cex=2)
  savePlot(filename=paste("Rplot", i), type="bmp")
}
cat(paste("Plots saved to", getwd()), "\n")
# GIF Construction Set
# Animation wizard
# add the bitmaps and save as gif animation file
# GIMP file open as layers ... save as animated gif

## End(Not run)

```

chi

*Chi-Squared Test***Description**

Performs chi-squared contingency table tests with informative output.

**Usage**

```
chi(x, rpct = 0, print = TRUE, plot = TRUE)
```

**Arguments**

x                      A numeric matrix.

rpct	A numeric scalar indicating the rounding used for printed output, default 0.
print	A logical scalar indicating if output should be printed, default TRUE.
plot	A logical scalar indicating if plot should be generated, default TRUE.

**Value**

A list with class "htest" containing the components described in [chisq.test](#).

**See Also**

[chisq.test](#).

**Examples**

```
## From Agresti(2007) p.39
M <- as.table(rbind(c(762, 327, 468), c(484, 239, 477)))
dimnames(M) <- list(gender=c("M", "F"),
  party=c("Democrat", "Independent", "Republican"))
chi(M)
```

---

CI

---

*Confidence Interval of Mean*


---

**Description**

Calculate the confidence interval or limits of the mean using the t distribution.

**Usage**

```
CI(x, alpha = 0.05, keepMean = TRUE, limits = TRUE, prefix = "",
  na.rm = FALSE)
```

**Arguments**

x	Numeric vector of sample observations.
alpha	Numeric scalar denoting significance level, default 0.05.
keepMean	Logical scalar indicating if the mean should be returned with the confidence interval/limits, default TRUE.
limits	Logical scalar indicating if the limits should be returned (otherwise a single interval is returned), default TRUE.
prefix	Character scalar to be used in assigning names to the returned value, default "".
na.rm	Logical scalar indicating if missing values should be removed before calculations, default FALSE.

**Value**

A named vector of the mean (if keepMean=TRUE) and the (1 - alpha)\*100 interval (if limits=FALSE). Names are the prefix concatenated to "mean", "lo", and "hi".

**Examples**

```
CI(1:10)
```



circles

*Draw Circles on a Plot***Description**

Draw circles on a plot, with control over circle size.

**Usage**

```
circles(x, y, z, data.range = range(z, na.rm = TRUE),
        circle.size.range = c(0.1, 1), outx = NA, outy = NA, add = FALSE,
        xlim = NULL, ylim = NULL, ...)
```

**Arguments**

<code>x</code>	Numeric vector of x coordinates.
<code>y</code>	Numeric vector of y coordinates.
<code>z</code>	Numeric vector of data used to inform the radii of the drawn circles. In general, this should be the square root of the quantity that you want to represent, because viewers will tend to judge the circles by their area not their radii.
<code>data.range</code>	Numeric vector, length 2, minimum and maximum <code>z</code> data to plot, default <code>range(z, na.rm=TRUE)</code> .
<code>circle.size.range</code>	Numeric vector, length 2, minimum and maximum circle radii in inches, default 0.1 to 1.
<code>outx</code>	Numeric scalar of x coordinate beyond the figure margins, default NA (see details).
<code>outy</code>	Numeric scalar of y coordinate beyond the figure margins, default NA (see details).
<code>add</code>	Logical scalar specifying if circles are added to existing plot (TRUE), default FALSE (a new plot is created).
<code>xlim</code>	Numeric vector, length 2, x-axis limits, unused if <code>add=TRUE</code> .
<code>ylim</code>	Numeric vector, length 2, y-axis limits, unused if <code>add=TRUE</code> .
<code>...</code>	Additional parameters supplied to the <a href="#">symbols</a> function.

**Details**

The size of the circles plotted corresponds directly with the range of data. For example, if there is a `z` of size `data.range[1]`, it will be plotted as a circle with radius `circle.size.range[1]`, and if there is a `z` of size `data.range[2]`, it will be plotted as a circle with radius `circle.size.range[2]`.

The default of NA for `outx` and `outy` places unseen smallest and biggest circles at a location where the x and y coordinates are 10 times the range observed plus the maximum observed. In most instances this should be well beyond the figure margins.

**Value**

A data frame with the name, class, dimension, and size of each member of the environment.

**Examples**

```
circles(trees$Height, trees$Girth, sqrt(trees$Volume),
data.range=sqrt(c(0, max(trees$Volume))), circle.size.range=c(0, 0.3),
xlab="Height (ft)", ylab="Diameter (in)", main="Tree Volume")
```

cleanup

*Quick Clean Up***Description**

Removes all objects from the current working directory.

**Usage**

```
cleanup()
```

**Details**

User is prompted for response. If the first letter of the response is "y" or "Y", all objects are removed from the current working directory.

colr

*Create a Range of Colors***Description**

Create a range of colors between two specified colors.

**Usage**

```
colr(x, fromcolname, tocolname)
```

**Arguments**

x	A numeric vector, the values to be assigned colors.
fromcolname	A character or numeric scalar, indicating the color to use for the lowest value in x. Either a color name (as listed by <a href="#">colors()</a> ), a hexadecimal string of the form "#rrggbb" or "#rrggbbaa" (see <a href="#">rgb</a> ), or a positive integer i meaning <a href="#">palette()[i]</a> .
tocolname	A character or numeric scalar, indicating the color to use for the highest value in x. See fromcolname.

**See Also**

[colors](#), [palette](#).

**Examples**

```
x <- 1:10
plot(x, x, pch=16, col=colr(x, "blue", "yellow"), cex=4)
```

coordflip

*Flip Coordinates***Description**

Flip (or reflect) coordinates across any given line.

**Usage**

```
coordflip(pts, seg)
```

**Arguments**

pts	A numeric matrix with two columns of x and y coordinates to be rotated.
seg	A numeric matrix of dimension 2 x 2 giving two points which define the line over which the coordinates will be flipped.

**Value**

A numeric matrix with same dimension as pts with the flipped x and y coordinates.

**References**

Based on a method posted by Il-Bhima on 22 July 2010 on stackoverflow [\[link\]](#).

**See Also**

[coordplot](#), [coordmove](#), [coordturn](#), [coordtri](#).

**Examples**

```
# starting coordinates
test <- matrix(c(0, 4, 1, 0, 2, 3), ncol=2,
  dimnames=list(LETTERS[1:3], NULL))
coordplot(test)
# flip the coordinates across the line defined by the first two points
ftest <- coordflip(test, test[1:2, ])
coordplot(ftest)
```

coordmove

*Move Coordinates***Description**

Move coordinates in the x and y direction in the plane.

**Usage**

```
coordmove(pts, from, to)
```

**Arguments**

pts	A numeric matrix with two columns of x and y coordinates to be moved.
from	A numeric vector of length 2, the starting x and y coordinates of a point.
to	A numeric vector of length 2, the ending x and y coordinates of a point.

**Value**

A numeric matrix with same dimension as pts with the moved x and y coordinates.

**See Also**

[coordplot](#), [coordturn](#), [coordflip](#), [coordtri](#).

**Examples**

```
test <- matrix(c(0, 4, 1, 0, 2, 3), ncol=2,
  dimnames=list(LETTERS[1:3], NULL))
coordplot(test)
# move the coordinates so that the second point is at the origin
mtest <- coordmove(test, test[2, ], c(0, 0))
coordplot(mtest)
```

---

coordplot

*Plot Coordinates*

---

**Description**

Quick plot of coordinates on equally scaled coordinate plane, labelled with row names.

**Usage**

```
coordplot(pts)
```

**Arguments**

pts	A numeric matrix with two columns of x and y coordinates to be plotted.
-----	---

**See Also**

[coordturn](#), [coordmove](#), [coordflip](#), [coordtri](#).

**Examples**

```
test <- matrix(c(0, 4, 1, 0, 2, 3), ncol=2,
  dimnames=list(LETTERS[1:3], NULL))
coordplot(test)
```

coordtri

*Determine Third Coordinate of Triangle***Description**

Determine the third coordinate of a triangle, given the other two coordinates and the distances to the third.

**Usage**

```
coordtri(cleft, cright, lleft, lright)
```

**Arguments**

cleft	A numeric vector of length 2, the x and y coordinates of the lower left point of the triangle (see details).
cright	A numeric vector of length 2, the x and y coordinates of the lower right point of the triangle (see details).
lleft	A numeric scalar, the length of the left side of the triangle, from cleft to the top (see details).
lright	A numeric scalar, the length of the right side of the triangle, from cright to the top (see details).

**Details**

The directions "left" and "right" refer to the triangle not necessarily in its current state, but rather **after** it has been rotated such that the bottom of the triangle is level (e.g., on the  $y=0$  line), and the third coordinate that the function returns is "above" the bottom (e.g.,  $y>0$ ).

A warning message is given if a triangle can't be built from the inputs provided, and the returned value is `c(NA, NA)`.

**Value**

A numeric vector of length 2, the x and y coordinates of the top point of the triangle.

**See Also**

[coordplot](#), [coordturn](#), [coordmove](#), [coordflip](#).

**Examples**

```
# define coordinates of base of triangle
AB <- rbind(A=c(-2, 4), B=c(2, -4))
# determine coordinates of top of triangle
# given base coordinates and side lengths
C <- coordtri(cleft=AB[1, ], cright=AB[2, ], lleft=12, lright=10)
# plot results
coordplot(rbind(C, AB))
```

coordturn

*Rotate Coordinates***Description**

Rotate coordinates clockwise around a pivot point.

**Usage**

```
coordturn(pts, pvt, rot)
```

**Arguments**

pts	A numeric matrix with two columns of x and y coordinates to be rotated.
pvt	A numeric vector of length 2, the x and y coordinates of the pivot point around which pts will be rotated.
rot	A numeric scalar indicating the amount of clockwise rotation, in radians.

**Value**

A numeric matrix with same dimension as pts with the rotated x and y coordinates.

**References**

Modification of code posted by Sage on 3 March 2011 on the website benn.org [\[link\]](#).

**See Also**

[coordplot](#), [coordmove](#), [coordflip](#), [coordtri](#).

**Examples**

```
# starting coordinates
test <- matrix(c(0, 4, 1, 0, 2, 3), ncol=2,
dimnames=list(LETTERS[1:3], NULL))
coordplot(test)
# rotate the coordinates clockwise 45 degrees
# around the first point (the origin)
rotest <- coordturn(test, test[1, ], rot=pi/4)
coordplot(rotest)
# rotate the coordinates counterclockwise 45 degrees
# around the first point (the origin)
rotest <- coordturn(test, test[1, ], rot=-pi/4)
coordplot(rotest)
```

---

`counties`*United States County Boundaries Map*

---

**Description**

Geographical data base for the county boundaries of the US, generated from US Department of the Census data.

**Format**

A list (of class map) with elements `x`, `y`, `range`, and `names`. The `x` and `y` vectors have longitude (`x`) and latitude (`y`) coordinates of successive county polygons, separated by NAs.

**Source**

A copy of the county data base from the `maps` package.

---

`dfclip`*Read Data Frame from Clipboard*

---

**Description**

Read in a data frame from text pasted to the clipboard.

**Usage**

```
dfclip(...)
```

**Arguments**

`...` Additional parameters to [read.table](#).

**Value**

Data frame.

**See Also**

[read.table](#).

---

doy	<i>Day of Year</i>
-----	--------------------

---

**Description**

Calculates the day of the year.

**Usage**

```
doy(date, day1 = "01-01")
```

**Arguments**

date	A vector of dates to be converted.
day1	A character scalar specifying what date should equate to day 1, use "01-01", the default, to get the Julian day.

**Value**

A numeric vector the same length as date giving the day of the year.

**Examples**

```
x <- as.Date(c("1963-01-15", "1972-07-20", "1999-03-10"))
doy(x)
doy(x, "03-01")
```

---

drawcell	<i>Draw Table Cell</i>
----------	------------------------

---

**Description**

Draw a table cell at location (r,c).

**Usage**

```
drawcell(title, r, c, text.cex = 1, bg.col = "white", frame.cell = TRUE)
```

**Arguments**

title	A character scalar, the text to be placed in the table cell.
r	An integer scalar, the row number of the table cell.
c	An integer scalar, the column number of the table cell.
text.cex	A numeric scalar, the relative size of the text to be place in the table cell, default 1.
bg.col	A character or numeric scalar, indicating the background color to use for the table cell, default "white".
frame.cell	A logical scalar, indicating whether a black border should be drawn around the table cell, default TRUE.



## Details

This is part of group of function to plot tables, modifications of functions in the Systmatic Investor Toolbox.

## Value

A table cell is added to the current table.

## References

Systmatic Investor Toolbox. <https://github.com/systematicinvestor/SIT>

---

fill	<i>Fill in Missing Values</i>
------	-------------------------------

---

## Description

Fill in missing values in a vector, using the last recorded value.

## Usage

```
fill(x, resetWhen = rep(FALSE, length(x)))
```

## Arguments

x	A vector, can be character, numeric, or logical.
resetWhen	A logical vector, the same length as x, indicating elements that should not be filled in.

## Details

Similar to [na.locf](#) in the zoo package, but works for "" in character vectors as well.

## Value

A vector the same length as x, with all NAs or ""s replaced by the last value for the vector. Note that and missing values at the beginning of the vector will not be replaced.

## Examples

```
numvec <- c(NA, 1:5, NA, NA, NA, 10:12, NA)
newgroup <- c(1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0)
fill(numvec)
fill(numvec, newgroup)

charvec <- c("", letters[1:5], "", "", "", letters[10:12], "")
fill(charvec)
```

---

first	<i>Identify the First Elements of Series of Repeated Values</i>
-------	---

---

**Description**

Identify the first elements of series of repeated values.

**Usage**

```
first(x)
```

**Arguments**

x	A vector whose values will be explored for series of repeated values, can be character, numeric, or factor.
---	---

**Value**

An integer vector the same length as x, with a 1 for every element that is different than the one before it, and a 0 for every element that is the same as the one before it.

**Examples**

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

---

formatdf	<i>Format a Data Frame</i>
----------	----------------------------

---

**Description**

Format selected columns of a data frame for pretty viewing or printing.

**Usage**

```
formatdf(df, numercol = NULL, round2 = 0, ndec = round2, comma = ",",
  characol = NULL, align = "left", keepnum = TRUE)
```

**Arguments**

df	Data frame to be formatted.
numercol	Numeric vector, index of numeric columns to format, default, NULL, uses all numeric and integer columns.
round2	Integer vector, indicating what place numbers should be rounded to, default 0.
ndec	Integer vector, indicating how many places to the right of the decimal point should be displayed, default round2.
comma	Character vector, indicating thousands character separator, default ",".
characol	Numeric vector, index of character columns to format, default, NULL, uses all character and factor columns.

align	Character vector, indicating justification of character columns, default "left". Other options include "right", "centre", and "none".
keepnum	Logical scalar, indicating if numeric columns should be kept as numeric, default TRUE, or converted to character.

### Details

If a vector of length 1 is provided for round2, ndec, comma, or align, the same rounding, decimals, commas, or alignment is applied to all specified columns.

### Value

A data frame, the same dimensions as df with selected columns rounded and/or formatted and converted to character.

### See Also

[format](#).

### Examples

```
head(mtcars)
formatdf(head(mtcars))
```

---

getpackages

*Get Packages*

---

### Description

Installs (if necessary) and attaches the specified packages.

### Usage

```
getpackages(want)
```

### Arguments

want	A character vector of package names.
------	--------------------------------------

---

inrange	<i>In Range</i>
---------	-----------------

---

**Description**

Test to see if value falls within specified range.

**Usage**

```
inrange(x, r, open = TRUE)
```

**Arguments**

x	A numeric vector of values to be tested.
r	A numeric vector of length 2 specifying the range.
open	A logical scalar indicating if the range interval is open (excludes endpoints, TRUE, default) or closed (includes endpoints, FALSE).

**Value**

A logical vector, the same length as x indicating if values fall within the specified range.

**Examples**

```
inrange(4:6, c(4, 5.5))
inrange(4:6, c(4, 5.5), FALSE)
```

---

jvaFirst	<i>Startup</i>
----------	----------------

---

**Description**

One function with all the commands I typically want run at the start of an R session.

**Usage**

```
jvaFirst(maxp = 500, ndec = 10, cont = "... ",
  mirror = "http://streaming.stat.iastate.edu/CRAN/", helpt = "html",
  fac = FALSE)
```

**Arguments**

maxp	Integer scalar, maximum number of lines printed, default 500.
ndec	Integer scalar, maximum width of fixed notation before switching to scientific notation, default 10.
cont	Character scalar, prompt used for lines which continue past first command line, default "... ".
mirror	Character scalar, CRAN mirror, default "http://streaming.stat.iastate.edu/CRAN/".
helpt	Character scalar, type of help, default "html".
fac	Logical scalar, use factors rather than character strings, default FALSE.

**See Also**

[options](#), [help](#).

**Examples**

```
jvaFirst()
```

---

jvaLast	<i>Shutdown</i>
---------	-----------------

---

**Description**

One function with all the commands I typically want run at the end of an R session.

**Usage**

```
jvaLast(file = ".Rhistory", nlines = 10000)
```

**Arguments**

file	Character scalar, file in which to save the commands history relative to current working directory, default ".Rhistory".
nlines	Integer scalar, number of lines that saved to commands history, default 10,000.

**See Also**

[savehistory](#), [Sys.setenv](#).

---

jvamic	<i>A Collection of Utility Functions</i>
--------	--

---

**Description**

**jvamic** is a collection of miscellaneous utility functions. Some of the more commonly used functions include [plotdf](#) to plot each variable of a data frame, [shadepoly](#) to add a shaded polygon of confidence intervals to a plot, and [AICc](#) to compare models using Akaike's Information Criterion.

---

jvamicenv	<i>jvamic Package Local Environment</i>
-----------	---

---

**Description**

An environment local to the jvamic package, used to hold objects outside of the individual package functions

**Format**

An environment.

**Source**

Post from Hadley Wickham to r-help on 2 Dec 2014 [\[link\]](#).

---

jvanames

*Format Names*


---

### Description

Format names to be all lower case, unique, and without underscores.

### Usage

```
jvanames(charvec)
```

### Arguments

charvec            A character vector to be trimmed.

### Value

A character vector, the same length as charvec, but with all lower case, unique, and without underscores.

### See Also

[make.names](#)

### Examples

```
jvanames(c("Conc_mgL", "Conc_mgL", "Temp"))
```

---

jvatree

*Cross Validated Tree Model*


---

### Description

Fit a classification or regression tree model to data, and automatically prune the tree based on the cross-validation error.

### Usage

```
jvatree(...)
```

### Arguments

...            Arguments provided to [rpart](#) including formula (the only required argument).

### Details

First a tree model is fit. Then the model is pruned such that it has the fewest number of splits that yield a mean cross-validation error less than the minimum cross-validation error plus one standard error.

**Value**

An object of class `rpart`. See [rpart.object](#).

**Examples**

```
fit <- jvatree(Ozone ~ ., data=airquality)
par(xpd=NA) # otherwise on some devices the text is clipped
plot(fit)
text(fit, use.n=TRUE)
```

---

Lakeabbs	<i>Great Lakes Abbreviations</i>
----------	----------------------------------

---

**Description**

A vector with the first initials of the five Great Lakes.

**Format**

A character vector, length 5.

---

Lakenames	<i>Great Lakes Names</i>
-----------	--------------------------

---

**Description**

A vector with the names of the five Great Lakes.

**Format**

A character vector, length 5.

---

last	<i>Identify the Last Elements of Series of Repeated Values</i>
------	--

---

**Description**

Identify the last elements of series of repeated values.

**Usage**

```
last(x)
```

**Arguments**

**x** A vector whose values will be explored for series of repeated values, can be character, numeric, or factor.

**Value**

An integer vector the same length as `x`, with a 1 for every element that is different than the one after it, and a 0 for every element that is the same as the one after it.

**Examples**

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

---

latlong2

---

*Convert Latitude and Longitude to US State or County*


---

**Description**

Convert latitude and longitude coordinates to US states or US states and counties.

**Usage**

```
latlong2(pointsDF, to = c("state", "county")[1])
```

**Arguments**

<code>pointsDF</code>	A data frame in which the first two columns contain the longitude and latitude in degrees.
<code>to</code>	A character scalar identifying the region to be identified, either "state", the default, or "county".

**Value**

If `to="state"`, a character vector of state names, all lower case, the same length as the number of rows in `pointsDF`. If `to="county"`, a character matrix of state and county names, all lower case, with two columns and the same number of rows as `pointsDF`.

**References**

Based on a method posted by Josh O'Brien on 6 Jan 2012 on stackoverflow [\[link\]](#).

**Examples**

```
testPoints <- data.frame(x=c(-90, -120), y=c(44, 44), z=c("a", "b"))
latlong2(testPoints, to="state")
latlong2(testPoints, to="county")
```



---

lls	<i>List All Members of an Environment.</i>
-----	--

---

**Description**

Generate a descriptive list of all members of an environment.

**Usage**

```
lls(pos = 1, pat = "")
```

**Arguments**

pos	A scalar of the environment, search path index, or package of interest, default 1.
pat	Regular expression pattern specifying which members to return, default "" for all members.

**Value**

A data frame with the name, class, dimension, and size of each member of the environment.

**References**

Based on a method posted by Bendix Carstensen on 10 Jan 2007 on R-help [\[link\]](#).

**Examples**

```
## Not run:
lls()

## End(Not run)
```

---

map5	<i>Great Lakes Basin Shoreline</i>
------	------------------------------------

---

**Description**

A single data frame with longitude and latitude coordinates of the shorelines in the Great Lakes basin.

**Format**

One row for each point, consecutive rows with non-missing values for each line.

**Author(s)**

Rich Signell, USGS.

**Source**

NOAA National Geophysical Data Center, (the now defunct) Coastline Extractor, [ngdc.noaa.gov/mgg/coast](http://ngdc.noaa.gov/mgg/coast).

---

mapL	<i>Great Lakes Shorelines</i>
------	-------------------------------

---

**Description**

List of five data frames, one for each Great Lake (Superior, Michigan, Huron, Erie, Ontario) with longitude and latitude coordinates of the shorelines.

**Format**

One row for each point, consecutive rows with non-missing values for each line.

**Author(s)**

Rich Signell, USGS.

**Source**

NOAA National Geophysical Data Center, (the now defunct) Coastline Extractor, [ngdc.noaa.gov/mgg/coast](https://ngdc.noaa.gov/mgg/coast).

---

mapSMR	<i>St. Marys River Shoreline</i>
--------	----------------------------------

---

**Description**

Data frame with longitude and latitude coordinates of the St. Marys River shoreline (the connecting channel between Lakes Superior and Huron in the Great Lakes basin).

**Format**

One row for each point, consecutive rows with non-missing values for each line.

**Author(s)**

Rich Signell, USGS.

**Source**

NOAA National Geophysical Data Center, (the now defunct) Coastline Extractor, [ngdc.noaa.gov/mgg/coast](https://ngdc.noaa.gov/mgg/coast).

---

matrixtrim	<i>Trim Matrix</i>
------------	--------------------

---

### Description

Trim the rows and columns of a logical matrix until every row and columns has a specified level of TRUE values.

### Usage

```
matrixtrim(m, prop = c(1, 1), rowsfirst = TRUE)
```

### Arguments

<code>m</code>	A logical matrix to be trimmed, should be all TRUE, FALSE (or 0, 1).
<code>prop</code>	A numeric vector of length two, the minimum proportion of TRUE values in each row and column of the trimmed matrix, default <code>c(1, 1)</code> .
<code>rowsfirst</code>	A logical scalar, indicating if rows (TRUE, the default) or columns (FALSE) should be trimmed from the matrix first.

### Value

A list of length three:

- `trim`=a logical matrix, the resulting trimmed matrix
- `dim`=a numeric vector of length 2, the dimensions of `trim`
- `n`=a numeric scalar, the total number of TRUE values in `trim`

### Examples

```
m <- matrix(rep(c(1, 0, 1, 0), c(4, 1, 13, 2)), nrow=5)
matrixtrim(m)
matrixtrim(m, rowsfirst=FALSE)
matrixtrim(m, prop=c(0.7, 0.7))
```

---

modecontin	<i>Mode of Continuous Variable</i>
------------	------------------------------------

---

### Description

Estimates the mode of a continuous variable.

### Usage

```
modecontin(x, plot = FALSE, ...)
```

**Arguments**

<code>x</code>	A numeric vector.
<code>plot</code>	A logical scalar indicating whether to plot the results, default FALSE.
<code>...</code>	Additional arguments to the <a href="#">density</a> function.

**Value**

The estimated mode of `x`.

**References**

Based on a method posted by Peter Dalgaard on 29 Aug 2008 on [r-help](#).

**Examples**

```
x <- rnorm(100)
modecontin(x, TRUE)
```

---

mrc

*Estimates from Mark-Recapture Studies*


---

**Description**

Estimates the recovery rate and abundance of a closed population from a mark-recapture study with a single capture and marking event and a single recapture event.

**Usage**

```
mrc(m, r, c, alpha = 0.05)
```

**Arguments**

<code>m</code>	A numeric scalar, the number of individuals marked and released.
<code>r</code>	A numeric scalar, the number of marked individuals recaptured.
<code>c</code>	A numeric scalar, the number of individuals checked for marks.
<code>alpha</code>	A scalar, the desired confidence level, default 0.05.

**Value**

A named vector with the estimated recovery rate  $U$  with lower and upper  $100*(1-\alpha)\%$  confidence limits from [binomCI](#), and the number of observations  $N$  (the bias-corrected Petersen estimator from Chapman 1954 Table 3) with lower and upper  $100*(1-\alpha)\%$  confidence limits from the normal approximation of the Poisson for  $r$  (Ricker 1975).

**References**

- D. G. Chapman. 1954. The estimation of biological populations. *Annals of Mathematical Statistics* 25:1-15. [\[link\]](#).
- W. E. Ricker. 1975. Computation and interpretation of biological statistics of fish populations. *Fisheries Research Board of Canada Bulletin*. 191.

See Also

[binomCI](#)

Examples

```
mrc(100, 10, 1000)
```

---

mytable	<i>Cross Tabulation and Table Creation</i>
---------	--

---

Description

Build a contingency table of the counts at each combination of factor levels, incorporating missing values by default.

Usage

```
mytable(...)
```

Arguments

... Arguments provided to [table](#).

Value

An array of integer values of class "table".

Examples

```
mytable(c(1, 1, 1, 2, NA, 3, 4, 1, 10, 3))
```

---

numbers2words	<i>Convert Integers to Words</i>
---------------	----------------------------------

---

Description

Convert integers into English words.

Usage

```
numbers2words(x, billion = c("US", "UK")[1], and = if (billion == "US") ""  
  else "and")
```

Arguments

- |         |  |
|---------|--|
| x       | An integer vector to be converted.   |
| billion | A character scalar indicating if "US" (default) or "UK" meaning of billion should be used.               |
| and     | A character scalar for use as a conjunction, set to "" when billion=="US" (default) and "and" otherwise. |

**Value**

A character vector, the same length as x giving the English word(s) for the integer(s) x.

**References**

John Fox. 2005. How Do You Spell That Number? Rnews Volume 5(1):51-54. [\[link\]](#).

**Examples**

```
numbers2words(c(456000000, -123, 1000, 12))
```

---

pkgin	<i>Install a Package</i>
-------	--------------------------

---

**Description**

Install a package from local files, load the library, and save installed package to zip archive.

**Usage**

```
pkgin(package, ld = "C:/Users/jvadams/Documents/R/win-library", lv = "max",  
pd = "C:/JVA/GitHub")
```

**Arguments**

- |         |   |
|---------|---|
| package | A character scalar, package name.   |
| ld      | A character scalar, R library directory, default "C:/Users/jvadams/Documents/R/win-library/3.2".            |
| lv      | A character scalar, R library directory version number, default "max", which uses the latest version in ld. |
| pd      | A character scalar, R package directory, default "C:/JVA/GitHub".   |

**See Also**

[pkgup](#)

---

pkgman	<i>Copy a Package Manual</i>
--------	------------------------------

---

**Description**

Copy a package reference manual created by R CMD Check to the corresponding GitHub folder.

**Usage**

```
pkgman(package, file = paste0("C:/Users/jvadams/", package, ".Rcheck/",  
package, "-manual.pdf"), dir = "C:/JVA/GitHub/")
```

**Arguments**

package	A character scalar, package name.
file	A character scalar, path of file to be copied, default paste0("C:/Users/jvadams/", package, ".Rcheck/", package, "-manual.pdf").
dir	A character scalar, directory where package folder is located, default "C:/JVA/GitHub/".

**See Also**

[pkgin](#)

---

pkgup	<i>Document a Package</i>
-------	---------------------------

---

**Description**

Document a package from local files.

**Usage**

```
pkgup(package, dir = "C:/JVA/GitHub/")
```

**Arguments**

package	A character scalar, package name.
dir	A character scalar, directory where package folder is located, default "C:/JVA/GitHub/".

**See Also**

[pkgin](#)

---

plotblank	<i>Create a Blank Plot</i>
-----------	----------------------------

---

**Description**

Create a blank plot (no symbols) on which to add other plotting features.

**Usage**

```
plotblank(x = 0:1, y = 0:1, xlab = "", ylab = "", las = 1, ...)
```

**Arguments**

x	A numeric vector, the x coordinates of points in the plot.
y	A numeric vector, the y coordinates of points in the plot.
xlab	A character scalar, title for the x axis, default "".
ylab	A character scalar, title for the y axis, default "".
las	A numeric scalar, style of axis labels, 0=always parallel to the axis, 1=always horizontal (default), 2=always perpendicular to the axis, 3=always vertical.
...	Additional arguments to the <a href="#">plot</a> function.

**See Also**

[plot](#), [title](#), [par](#)

**Examples**

```
plotblank(xlim=c(1, 100))
```

---

plotcor

*Plot a Matrix of Correlations*

---

**Description**

Plot a matrix of correlations using white ellipses (representing the measured correlation) overlaid on colored circles.

**Usage**

```
plotcor(r, addtext = TRUE, atcex = NULL, incdiag = FALSE, rorder = TRUE,
        plot = TRUE, ...)
```

**Arguments**

<code>r</code>	A matrix of correlations (need not be symmetrical) with values ranging from -1 to 1.
<code>addtext</code>	A logical scalar indicating whether the value of each correlation should be written over the ellipses, default TRUE.
<code>atcex</code>	A numeric scalar giving the magnification of the added ellipse text relative to that set in <a href="#">par</a> , default NULL results in <code>atcex=8/maximum dimension of r</code> .
<code>incdiag</code>	A logical scalar indicating whether to plot circles for the diagonal values of the matrix (if symmetric), default FALSE.
<code>rorder</code>	A logical scalar indicating whether the columns and rows of the matrix should be reordered using seriation, default TRUE.
<code>plot</code>	A logical scalar indicating whether the plot should be drawn, default TRUE.
<code>...</code>	Additional parameters to <a href="#">par</a> .

**Details**

Correlation is represented as a white ellipse over a colored circle, sized so that the proportion of the colored circle visible beyond the ellipse is equal to the squared correlation ( $r^2$ ). The color of each circle ranges from cyan (for  $r = -1$ ) to magenta (for  $r = 1$ ) through white (for  $r = 0$ ). Similarly, the transparency of each printed correlation value (if `addtext=TRUE`) ranges from 1 (for  $r = 0$ ) to 0 (for  $|r| = 1$ ).

**Value**

A list of with two vector of integers (the same length as each dimension of `r`) representing the linear order suggested by seriation.



**See Also**

[plotcorr](#) on which the idea for the function was based, [seriate](#), and [draw.ellipse](#).

**Examples**

```
# example using a symmetric matrix
sr <- cor(swiss)
sord <- plotcor(sr)
sr[sord[[1]], sord[[2]]]
# example using an asymmetric matrix
lr <- cor(longley)[1:3, 4:7]
lord <- plotcor(lr)
lr[lord[[1]], lord[[2]]]
```

---

plotdf

*Plot Data Frame*


---

**Description**

Plots each variable of a data frame.

**Usage**

```
plotdf(df, ...)
```

**Arguments**

df	A data frame.
...	Other parameters to <code>par(...)</code> .

**Value**

A data frame with summary statistics for each variable in the data frame, number entered, number missing, number unique, minimum, mean, maximum, and the ratio of the maximum over the non-zero minimum.

**See Also**

[par](#).

**Examples**

```
plotdf(mtcars)
```

---

predAntilog	<i>Unbiased Prediction of Log Transformed Response on Original Scale</i>
-------------	--

---

**Description**

Provide unbiased estimates on the original scale from an analysis of variance model with a log transformed response.

**Usage**

```
predAntilog(aovfit, xdata, logbase = exp(1), k = 0)
```

**Arguments**

aovfit	An object of class c("aov", "lm").
xdata	A data frame with predictor variables corresponding to those in model for which predictions should be made.
logbase	A numeric scalar, the base of the log transformation used in the transformed response of model, default exp(1).
k	A numeric scalar, the constant added to the response prior to transformation, default 0.

**Value**

A numeric vector of predicted values on the original scale of the response.

**Examples**

```
fit <- aov(log(yield) ~ block + N * P + K, npk)
predAntilog(fit, npk)
```

---

prettylog	<i>Pretty Breakpoints on Log Scale</i>
-----------	--

---

**Description**

Compute a sequence of "round" values which cover the range of x on the log scale.

**Usage**

```
prettylog(x, lead = c(1, 5), extra = 5)
```

**Arguments**

x	A numeric vector.
lead	An integer vector giving the desired lead digits of pretty values on the log scale, default c(1, 5).
extra	An integer scalar giving the desired number of additional non-log scale values to include, default 5.

**Value**

A numeric vector of pretty values covering the range of x on the log scale.

**Examples**

```
vals <- rlnorm(100, 6)
summary(vals)
prettylog(vals, 1, 0)
prettylog(vals, 1)
prettylog(vals, c(1, 2, 5))
```

---

prettytable	<i>Prettify the Numeric Columns of a Table</i>
-------------	--

---

**Description**

Prettify the numeric columns of a table, by formatting them and converting them to character columns for printing.

**Usage**

```
prettytable(df, digits = 2, rounds = TRUE, bigseps = ",")
```

**Arguments**

df	A data frame to be prettified.
digits	Integer vector of either length 1 or the number of columns in df, number of digits to be used, default 2. See round.
rounds	Logical vector of either length 1 or the number of columns in df, indicating whether numbers should be rounded to digits decimal places (TRUE, the default), rounded to digits significant digits (FALSE), or not rounded at all (NULL).
bigseps	Character vector of either length 1 or the number of columns in df, giving the character to be used as a mark between every three digits before the decimal, default ",".

**Value**

A data frame the same dimensions as df with the numeric columns converted to character columns, formatted as specified.

**Examples**

```
head(mtcars)
prettytable(head(mtcars))
```

---

ratest	<i>Ratio Estimation</i>
--------	-------------------------

---

## Description

Ratio estimation.

## Usage

```
ratest(num, den, adj = 1)
```

## Arguments

num	A numeric vector, the numerator of the ratio.
den	A numeric vector, the denominator of the ratio.
adj	A numeric scalar, an adjustment factor to be multiplied by the ratio, default 1.

## Details

All records with a missing value in either the numerator or the denominator are omitted from calculations.

## Value

A named vector with the sum of the numerators, the sum of the denominators, the ratio of the sums, the number of non-missing input pairs, and the standard deviation and 95

## See Also

[svydesign](#), [svyratio](#).

## Examples

```
# Size, weekly income, and food cost of 33 families from
# Cochran's (1977) Sampling Techniques.
familysize <- c(2, 3, 3, 5, 4, 7, 2, 4, 2, 5, 3, 6, 4, 4,
2, 5, 3, 4, 2, 4, 2, 5, 3, 4, 7, 3, 3, 6, 2, 2, 6, 4, 2)
income <- c(62, 62, 87, 65, 58, 92, 88, 79, 83, 62, 63, 62, 60, 75, 90,
75, 69, 83, 85, 73, 66, 58, 77, 69, 65, 77, 69, 95, 77, 69, 69, 67, 63)
foodcost <- c(14.3, 20.8, 22.7, 30.5, 41.2, 28.2, 24.2, 30, 24.2, 44.4,
13.4, 19.8, 29.4, 27.1, 22.2, 37.7, 22.6, 36, 20.6, 27.7, 25.9, 23.5,
39.8, 16.8, 37.8, 34.8, 28.7, 63, 19.5, 21.6, 18.2, 20.1, 20.7)
# weekly expenditure on food per person
ratest(foodcost, familysize)
# percentage of income spent on food
ratest(foodcost, income, 100)
```

---

recode	<i>Recode Values</i>
--------	----------------------

---

### Description

Assign new values to a vector.

### Usage

```
recode(x, old, new, must.match = FALSE)
```

### Arguments

x	A vector whose values will be recoded, can be character, numeric, or factor.
old	A vector of the unique values currently in the vector.
new	A vector of values which should replace the current ones.
must.match	A logical scalar indicating whether only those elements of the original vector with values in old should be returned (TRUE), or all values should be returned (FALSE, default) though some may be unchanged.

### Value

A vector the same length as x (unless must.match=TRUE), with old values replaced by new values.

### Examples

```
recode(c(1,1,1,2,3,4,1,10,3), 1:3, 1001:1003)
recode(c(1,1,1,2,3,4,1,10,3), 1:3, 1001:1003, must.match=TRUE)
```

---

segreg	<i>Segmented Regression</i>
--------	-----------------------------

---

### Description

Fit a simple linear segmented regression (also known as changepoint, hockey stick, or broken line regression).

### Usage

```
segreg(x, y, k = NULL)
```

### Arguments

x	Numeric vector, the independent variable which will be "broken".
y	Numeric vector, the dependent variable which will be used to define the break point.
k	Numeric scalar, the location of the break point, if known. If NULL, the default, the optimal break point will be chosen from among all unique values of x except for the minimum and the maximum.

## Details

The break point cuts the x vector into two groups,  $x \leq k$  and  $x > k$ .

## Value

List with three elements, fit: the resulting lm object, k, and pred: data frame with the unique values of x (and k) with predicted values.

## Examples

```
indvar <- sample(1:50, 100, replace=TRUE)
depvar <- ifelse(indvar<32, indvar + 3, indvar/4 + 27) + rnorm(100, sd=2)
sr <- segreg(indvar, depvar)
plot(indvar, depvar)
lines(sr$pred)
abline(v=sr$k, lty=2)
```

---

shadepoly

Add Shaded Polygon to Plot

---

## Description

Add a spline-smoothed, shaded polygon to a plot, typically to show an interval or range of values (in the y-direction) for a time series of x values.

## Usage

```
shadepoly(x, ymd, ylo, yhi, subset = NULL, kol = "#000000", opq = c(20,
50), addline = TRUE)
```

## Arguments

x	A numeric vector, the metric to plot in the x direction (e.g., time).
ymd	A numeric vector, the metric to plot in the y direction (e.g., a mean or median).
ylo	A numeric vector, the lower interval or range in the y direction (e.g., a lower confidence interval or quartile).
yhi	A numeric vector, the upper interval or range in the y direction (e.g., an upper confidence interval or quartile).
subset	A logical vector, indicating subset of the data to plot.
kol	A character scalar, the hex color to use for plotting both the shaded polygon and (if requested) the line, default "#000000" (black).
opq	A numeric vector of length 2, opacity for the polygon and the line, default c(20, 50).
addline	A logical scalar, indicating if the x vs ymd line should be added to the plot (on top of the shaded polygon), default TRUE.

## Details

Missing values are removed prior to plotting, such that there will be no breaks in the shaded polygon nor the line (if requested). The lower and upper intervals of the polygon are spline-smoothed prior to plotting, as is the line (if requested).

**See Also**

[polygon](#), [lines](#).

**Examples**

```
x <- 1:10
y <- sample(10)
noise <- abs(rnorm(10))
plot(x, y, ylim=range(y-noise, y+noise), type="n")
shadepoly(x, y, y-noise, y+noise)
```

---

showmarks	<i>Show Marks</i>
-----------	-------------------

---

**Description**

Show marks used in graphing, including line types, plotting symbols, default colors, color blind friendly colors ('blindcolz'), and fonts.

**Usage**

```
showmarks()
```

**Examples**

```
showmarks()
```

---

states	<i>United States State Boundaries Map</i>
--------	---

---

**Description**

Geographical data base for the state boundaries of the US, generated from US Department of the Census data.

**Format**

A list (of class map) with elements x, y, range, and names. The x and y vectors have longitude (x) and latitude (y) coordinates of successive state polygons, separated by NAs.

**Source**

A copy of the state data base from the maps package.

stringin

*Find a String in a Character Vector***Description**

Find a string in a character vector.

**Usage**

```
stringin(pattern, x, ignore.case = TRUE, value = TRUE, fixed = TRUE, ...)
```

**Arguments**

pattern	Character scalar, string to be matched in x.
x	Character vector where matches are sought.
ignore.case	Logical scalar, indicating case sensitivity, default TRUE.
value	Logical scalar, indicating whether to return the matching elements (the default, TRUE) or their indices (FALSE).
fixed	Logical scalar, indicating whether string should be matched as is, default TRUE.
...	Other arguments to <a href="#">grep</a> .

**Value**

Character vector containing the matching elements of x.

**See Also**

[grep](#).

**Examples**

```
txt <- c("The", "licenses", "for", "most", "software", "are", "designed",
"to", "take", "away", "your", "freedom", "to", "share", "and", "change",
"it.", "", "By", "contrast,", "the", "GNU", "General", "Public", "License",
"is", "intended", "to", "guarantee", "your", "freedom", "to", "share", "and",
"change", "free", "software", "--", "to", "make", "sure", "the", "software",
"is", "free", "for", "all", "its", "users.")
stringin("b", txt)
stringin("ar", txt)
```



---

trimspace	<i>Trim Whitespace</i>
-----------	------------------------

---

**Description**

Trim leading and trailing whitespace from a character vector.

**Usage**

```
trimspace(charvec)
```

**Arguments**

charvec            A character vector to be trimmed.

**Value**

A character vector, the same length as charvec, without leading and trailing whitespace.

**Examples**

```
trimspace("  Bob  and Alice")
trimspace(" Harriet and   June  ")
```

---

tweethead	<i>Tweet Headlines</i>
-----------	------------------------

---

**Description**

Tweet the latest headlines from the specified website.

**Usage**

```
tweethead(tweet = TRUE, username = NULL, website = NULL,
  credOrPath = "C:/JVA/R/Working Directory/.Renviro")
```

**Arguments**

tweet	A logical scalar indicating if tweets should be posted, default TRUE.
username	A character scalar, giving the name of the twitter user. The default, NULL, uses information stored in local .Renviro file (see Details).
website	A character scalar, giving the name of the website, from which to pull headlines. The default, NULL, uses information stored in local .Renviro file (see Details).
credOrPath	Either a character scalar giving the path of the environment where the credentials are stored or a character vector of length four, giving the credentials themselves: twitter_api_key, twitter_api_secret, twitter_access_token, and twitter_access_token_secret. The default is "C:/JVA/R/Working Directory/.Renviro" (see Details).

**Details**

This function is customized to work on a particular website. It's not for general use. To store information in local .Renviron file, use `writelnLines(c("username=xxx", "website=xxx", "twitter_api_key=xxx", "t`

**Value**

A named vector with the Mean, lower and upper confidence limits (L and U), and the number of observations N.

**References**

Simon Munzert. 19 Jan 2015. Programming a Twitter bot - and the rescue from procrastination. <http://www.r-datacollection.com/blog/Programming-a-Twitter-bot/>

Simon Munzert. 21 Dec 2014. How to conduct a tombola with R. [www.r-datacollection.com/blog/How-to-conduct-a-tombola-with-R/](http://www.r-datacollection.com/blog/How-to-conduct-a-tombola-with-R/)

**Examples**

```
## Not run:
tweethead(FALSE)
tweethead()

## End(Not run)
```

# Index

addedvar, 3  
addhist, 3  
AICc, 4, 29  
allcombs, 5  
  
bcpCI, 6  
binomCI, 7, 36, 37  
blindcolz, 7  
boot, 6  
brewcol, 8  
brewer.pal, 8  
  
calcr2, 8  
capwords, 9  
casefold, 9  
char2num, 10  
cheat, 11  
chi, 15  
chisq.test, 16  
CI, 16  
circles, 17  
cleanup, 18  
colors, 18  
colr, 18  
coordflip, 19, 20–22  
coordmove, 19, 19, 20–22  
coordplot, 19, 20, 20, 21, 22  
coordtri, 19, 20, 21, 22  
coordturn, 19–21, 22  
counties, 23  
  
density, 36  
dfclip, 23  
doy, 24  
draw.ellipse, 41  
drawcell, 24  
  
fill, 25  
first, 26  
format, 27  
formatdf, 26  
  
getpackages, 27  
grep, 48  
  
help, 29  
hist, 4  
  
inrange, 28  
  
jvaFirst, 28  
jvaLast, 29  
jvamic, 29  
jvamic-package (jvamic), 29  
jvamicenv, 29  
jvanames, 30  
jvatree, 30  
  
Lakeabbs, 31  
Lakenames, 31  
last, 31  
latlong2, 32  
lines, 47  
lls, 33  
  
make.names, 30  
map5, 33  
mapL, 34  
mapSMR, 34  
matrixtrim, 35  
modecontin, 35  
mrc, 36  
mytable, 37  
  
na.locf, 25  
numbers2words, 37  
  
options, 29  
  
palette, 18  
par, 40, 41  
pkgin, 38, 39  
pkgman, 38  
pkgup, 38, 39  
plot, 39, 40  
plotblank, 39  
plotcor, 40  
plotcorr, 41  
plotdf, 29, 41  
polygon, 47

predAntilog, [42](#)  
prettylog, [42](#)  
prettytable, [43](#)  
  
ratest, [44](#)  
read.table, [23](#)  
recode, [45](#)  
rgb, [18](#)  
rpart, [30](#)  
rpart.object, [31](#)  
  
savehistory, [29](#)  
segreg, [45](#)  
seriate, [41](#)  
shadepoly, [29](#), [46](#)  
showmarks, [47](#)  
states, [47](#)  
stringin, [48](#)  
svydesign, [44](#)  
svyratio, [44](#)  
symbols, [17](#)  
Sys.setenv, [29](#)  
  
table, [37](#)  
title, [40](#)  
trimspace, [49](#)  
tweethead, [49](#)