

# Package ‘RPMG’

September 23, 2019

**Type** Package

**Title** Graphical User Interface (GUI) for Interactive R Analysis Sessions

**Version** 2.2-3

**Date** 2019-09-20

**Author** Jonathan M. Lees [aut, cre],  
Jake Anderson [ctb]

**Maintainer** Jonathan M. Lees <jonathan.lees@unc.edu>

**Description** Really Poor Man's Graphical User Interface, used to create interactive R analysis sessions with simple R commands.

**License** GPL (>= 2)

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2019-09-23 10:30:02 UTC

## R topics documented:

RPMG-package . . . . .	2
aGETXprofile . . . . .	4
breakline.index . . . . .	5
butdoc . . . . .	7
chooser . . . . .	8
circle . . . . .	10
colwheel . . . . .	11
cprint . . . . .	12
endSCALE . . . . .	13
fmod . . . . .	14
Gcols . . . . .	15
getmem . . . . .	16
helpcolors . . . . .	17
HOZscale . . . . .	18
ilocator . . . . .	19

itoxyz . . . . .	20
jpg . . . . .	21
jpostscript . . . . .	22
label.it . . . . .	24
local.file . . . . .	24
meshgrid . . . . .	25
OPTREPLOT . . . . .	26
pastel.colors . . . . .	28
pickcolors . . . . .	29
rainbow.colors . . . . .	30
RESCALE . . . . .	30
rowBUTTONS . . . . .	31
see.pal . . . . .	33
SELOPT . . . . .	33
sepia.colors . . . . .	35
setXMCOL . . . . .	36
shade.col . . . . .	36
SHOWPAL . . . . .	37
slideshow . . . . .	38
textrect . . . . .	39
VVwheel . . . . .	41
wheelrgb . . . . .	42
whichbutt . . . . .	43
writeCOMMENT . . . . .	44
XSECDEM . . . . .	45
xyztoi . . . . .	47
ymarginfo . . . . .	48

**Index** **49**

---

RPMG-package	<i>Really Poor Man's GUI: sets up buttons for a graphical user interface in R</i>
--------------	---

---

**Description**

Package consists of two functions for setting up a GUI using only R-code.

**Details**

Package:	RPMG
Type:	Package
Version:	2.1-2
Date:	2008-08-15
License:	GPL

**Author(s)**

Jonathan M. Lees <jonathan.lees@unc.edu>

**See Also**

rowBUTTONS, whichbutt

**Examples**

```
### get sample image data set.
data(volcano)
##### set sample interval unit
attr(volcano, 'dx') =10
attr(volcano, 'dy') =10
### create the list of labels
### Actions for these buttons are described in the calling program XSECDEM
mybutts = c("DONE", "REFRESH", "rainbow", "topo", "terrain", "CONT", "XSEC", "PS" )
XSECDEM(volcano, mybutts)
#####
##### CODE STUB
## Not run: ### Example code chunk:
### general set up of RPGM usage:
##### make a plot
##### set buttons
buttons = rowBUTTONS(c("BUT1", "BUT2") , col=c(1,1), pch=c(1,1))
##### after plotting, locate in plot...
zloc = locator()
Nclick = length(zloc$x)
##### the last click on the screen before stopping (middle
##### mouse click) is used to set the action
K = whichbutt(zloc , buttons)
while(TRUE)
{
if(K[Nclick] == match("BUT1", labs, nomatch = NOLAB))
{
### do what ever button 1 is supposed to do
}
if(K[Nclick] == match("BUT2", labs, nomatch = NOLAB))
{
### do what ever button 2 is supposed to do
}
} ## end while loop

## End(Not run)
```

---

aGETXprofile

*Cross sectional profile through a digital elevation map*


---

### Description

Example of how to use RPMG button functions. This example shows how to plot a DEM and interactively change the plot and find projected cross-sections through a surface.

### Usage

```
aGETXprofile(jx, jy, jz, LAB = "A", myloc = NULL, PLOT = FALSE, asp=1)
```

### Arguments

jx, jy	locations of grid lines at which the values in 'jz' are measured.
jz	a matrix containing the values to be plotted
LAB	Alphanumeric (A-Z) for labeling a cross section
myloc	Out put of Locator function
PLOT	logical. Plot is created if TRUE
asp	aspect ration, see par

### Details

The program uses a similar input format as image or contour, with structure from the locator() function of x and y coordinates that determine where the cross section is to be extracted.

### Value

Returns a list of x,z values representing the projected values along the cross section.

RX	distance along cross section
RZ	values extracted from the elevation map

### Note

The program is an auxiliary program provided to illustrate the RPMG interactive R analysis.

### Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

### See Also

locator, image

**Examples**

```

## Not run:
##### get data
data(volcano)
#### extract dimensions of image
nx = dim(volcano)[1]
ny = dim(volcano)[2]

### establish units of image
jx = 10*seq(from=0, to=nx-1)
jy = 10*seq(from=0, to=ny-1)

#### set a letter for the cross section
LAB = LETTERS[1]

### coordinates of cross section on image
### this is normally set by using the locator() function
x1 = 76.47351
y1 = 231.89055
x2 = 739.99746
y2 = 464.08185

## extract and plot cross section

aGETXprofile(jx, jy, volcano, myloc=list(x=c(x1, x2), y=c(y1, y2)), LAB=LAB, PLOT=TRUE)

## End(Not run)

```

---

breakline.index      *Break a vector into segments*

---

**Description**

Break a vector into segments

**Usage**

```
breakline.index(Z, ww)
```

**Arguments**

Z	vector
ww	indices where the breaks should occur. if a matrix is provided the start and end indices are given, else the breaks are provided.

**Details**

Codes used for maps to break map segments along boundaries. But this is more general, nd can be used to break any vector according to given indices. See examples.

**Value**

List of indices that are segments.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```
### example with a vector of breaks
h = 1:20
k = breakline.index(h, c(8, 14))

##### select with a matrix of start-ends
r1 = rbind(c(3,10), c(14, 18))
k = breakline.index(h, r1)

j1 = seq(from=3, to=17, by=3)
j2 = j1+5

##### overlapping sequences
r1 = cbind(j1, j2)
k = breakline.index(h, r1)

##### example with coordinates

#### some data:
uu=list()
uu$x=c(136.66,136.34,136.07,136.07,135.62,135.03,134.98,
134.98,135.07,135.25,135.75,137.07,137.35,137.44,138.07,
138.07,137.80,137.75,137.25)
uu$y=c(39.878,39.749,39.490,39.296,39.200,39.135,38.909,
38.618,38.327,38.004,37.875,37.875,38.327,38.489,
38.812,39.006,39.232,39.587,39.943)

### plot raw data
plot(uu$x, uu$y, type="l")

#### cutoff:
z1 = 39

h = 1:length(uu$x)

w1 = which( uu$y>z1)

g1 = list(x=uu$x[w1] , y=uu$y[w1] )

lines(g1, col='red')
##### notice the connecting line.
##### how can we avoid this?
```

```
w2 = which(diff(w1)!=1)

k = breakline.index(w1, w2)

for(i in 1:length(k)) lines(uu$x[ k[[i]] ], uu$y[ k[[i]] ], col='blue')
##### see, line is broken correctly
```

---

butdoc

*Button Documentation for RPMG codes*

---

## Description

Interactive Button Documentation for RPMG codes

## Usage

```
butdoc(tag, doc, NEW = FALSE)
```

## Arguments

tag	character vector of tags
doc	character vector of (short) explanations
NEW	logical, TRUE = open new device

## Details

This is used in conjunction with interactive codes that employ RPMG

## Value

Side Effects

## Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

## See Also

chooser

**Examples**

```

ALLLABS = c( "DONE", "REFRESH", "EPS", "LINE", "DECIM", "MAP", "SURF", "TRACE", "TTC", "CITY", "TRcol",
             "STName", "Pick", "ZOOM", "UNZOOM", "IDARR", "FILT", "UnFILT", "P-GEN")
N = length(ALLLABS)
DOC = rep(NA, length=N)

DOC[1] = "Quick and return to calling program"
DOC[2] = "refresh screen"
DOC[3] = "Postscript plot"
DOC[4] = "draw a line (even number of clicks)"
DOC[5] = "Decimate the traces"
DOC[6] = "Make a map with great circles"
DOC[7] = "Draw a set of surface wave arrivals"
DOC[8] = "Toggle drawing of traces"
DOC[9] = "Travel Time Curves"
DOC[10] = "put random cities on X-axis"
DOC[11] = "toggle plotting traces with colors"
DOC[12] = "put station names on X-axis"
DOC[13] = "Pick arrivals on one trace"
DOC[14] = "Zoom display (need two clicks on screen)"
DOC[15] = "unzoom to original display"
DOC[16] = "Identify traces"
DOC[17] = "Fitler traces with a set of filters provided"
DOC[18] = "Unfilter traces to original display"
DOC[19] = "Run PICK.GEN on selected traces: select on the tags at X-axis"

butdoc(ALLLABS, DOC, NEW=FALSE)

```

---

 chooser

*Interactive Selection Winder*


---

**Description**

Choose an option from a selection

**Usage**

```

chooser(opts=c(1, 2, 5, 10, 15, 20) , ncol=5, nsel=NA,
        newdev=TRUE, STAY=FALSE,
        cols="red", main="", newplot=TRUE,
        xlim=c(0,1), ylim=c(0,1),
        just="CEN", ... )

```



**Arguments**

opts	list of options
ncol	number of columns
nselect	number of selections
newdev	logical, TRUE=start new device, default=TRUE
STAY	logical, TRUE=keep same device when done, default=FALSE
cols	colors for buttons, default = pastel.col(N)
main	title for screen (maybe instructions for picking)
newplot	logical, TRUE means start a new plot
xlim	xlim on the plot
ylim	ylim on the plot
just	character, justification in box, one of CEN, LEFT, RIGHT
...	additional parameters from par, used for font, cex, etc...

**Details**

Used for interactive selections of numeric or other options. If the input vector is all numeric, a numeric value is returned. If, on the other hand, the input is mixed or character, a character vector is returned. If the selection number nselect is left blank, it is set at 1. If it is specified, selection can be truncated by clicking the right mouse.

**Value**

vector of selections.

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

locator

**Examples**

```
## Not run:  
k = letters[1:26]  
  
pk = chooser(opts=k , nselect=3 )  
  
print(pk)  
  
k = c( 1:26, letters[1:26])  
pk = chooser(opts=k , nselect=3 )
```

```

print(pk)

k = 1:12

pk = chooser(opts=k , nsel=3 )

print(pk)
#####

plot(runif(10, 1, 100), runif(10, 1, 100), type='n')

APAL = c('tan2','red2','lightpink3','chocolate4','blue3','thistle4',
'lightcyan4',
'orangered1','purple4','darkred',
'dodgerblue1','gold3','chartreuse',
'sienna4')

##  nchar( APAL )
wm = which.max(nchar( APAL ))
swidth = strwidth(APAL[wm])

upar = par("usr")

mhgt = sum( strheight(APAL )+0.5*strheight(APAL ))

mwid = max( strwidth(APAL) )

mwid = mwid + 0.05*mwid

chooser(opts=APAL , ncol=1, nsel=NA, newdev=FALSE, STAY=TRUE,
newplot=FALSE, xlim=c(upar[1], upar[1]+mwid) ,
ylim=c( upar[4]-mhgt),upar[4]) , main="" )

## End(Not run)

```

---

circle

*circle coordinates*


---

### Description

generate circle coordinates for plotting

### Usage

```
circle(n = 1, ang1=0)
```

**Arguments**

n	number of points
ang1	starting angle (degrees)

**Value**

List	
x	coordinates
y	coordinates

**Author(s)**

Jonathan M. Lees <jonathan.lees@unc.edu>

**Examples**

```
j = circle(26)
plot(j)
```

---

colwheel	<i>Choose rgb from a color rectangle</i>
----------	--

---

**Description**

Shows and image of colors and allows one to choose a color and see what it looks like in swath with different backgrounds.

**Usage**

```
colwheel(v = 1, BACK = "black")
```

**Arguments**

v	v, from hsv color scheme
BACK	starting background color

**Value**

vector of RGB colors in hex format.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

hsv, VVwheel, wheelrgb, SHOWPAL.A

**Examples**

```
## Not run:  
colwheel(v = 1, BACK = "black")  
  
colwheel(v = 1, BACK = "white")  
  
## End(Not run)
```

---

cprint

*dump assignment*

---

**Description**

dump out an R assignemnt statement to the screen

**Usage**

```
cprint(a)
```

**Arguments**

a                   R object

**Value**

side effects

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```
x = 10  
  
cprint(x)
```

---

endSCALE                      *Plot nice scale at end of trace.*

---

**Description**

Calculate nice scale to use at the end of a plot. Use as an alternative to magicaxis.

**Usage**

```
endSCALE(arange, digits = 3)
```

**Arguments**

arange	2-vector of bounds
digits	number of digits to use

**Details**

The function returns information for plotting a nice bounds axis similar to MATLAB plotting style.

**Value**

character vector: min, max, exponent

**Note**

If the bounds span multiple orders of magnitude, may want to make adjustments (like setting a negative exponent bound to zero)

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

plotwlet

**Examples**

```
M = 1e-19
m = M

for(i in 1:10) {
  z = c( rnorm(1)*m ,  rnorm(1)*M )
  print(z)
  print( endSCALE(z) )
}

##### use in plotting:
```

```

x = seq(from=0, by=0.01, length=200)
a = 10000*rnorm(length(x))
old.par <- par(no.readonly = TRUE)
##### make room on the right margin
MAI = par("mai")
MAI[4] = MAI[2]
par(mfrow=c(2,1))
par(mai=MAI)
par(xaxs='i', yaxs='i')

plot(x,a, type='l')
  axtrace = range(a)
  Elabs = endSCALE(axtrace)
  exp = parse(text = Elabs[3])
axis(4, at=axtrace, labels=Elabs[1:2], pos=max(x), tick=TRUE, line=0.5, cex.axis=0.8, las=2)
  mtext(exp, side = 3, at = max(x), line=0.5, adj=-1, cex=0.8)
  mtext("m/s", side = 4, at =mean(axtrace), line=0.5, cex=0.8, las=1)

a = rnorm(length(x))/100000

plot(x,a, type='l')
  axtrace = range(a)
  Elabs = endSCALE(axtrace)
  exp = parse(text = Elabs[3])
axis(4, at=axtrace, labels=Elabs[1:2], pos=max(x), tick=TRUE, line=0.5, cex.axis=0.8, las=2)
  mtext(exp, side = 3, at = max(x), line=0.5, adj=-1, cex=0.8)
  mtext("m/s", side = 4, at =mean(axtrace), line=0.5, cex=0.8, las=1)

par(old.par)

}

```

---

fmod

*Floating point remainder function*


---

### Description

extract remainder for floating point numbers

### Usage

```
fmod(k, m)
```

**Arguments**

k	floating point number
m	divisor number

**Value**

returns remainder after dividing out the divisor part:  
j = floor(k/m)  
a = k-m\*j  
return(a)

**Author(s)**

Jonathan M. Lees <jonathan.lees@unc.edu>

**Examples**

```
### degrees after removing extraneous 2*pi  
j = 540.23  
fmod(j, 360)
```

---

Gcols

*Get Color Palette*

---

**Description**

Get Color Palette

**Usage**

```
Gcols(plow = 10, phi = 10, N = 100, pal = "rainbow", mingray = 0.5)
```

**Arguments**

plow	lowest number for color selection
phi	highest number for color selection
N	number of colors
pal	color palette name
mingray	lower end is blanked out and replaced by gray

**Value**

c(LOW, Z, HI) color palette

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

tomo.colors, shade.col

**Examples**

```
TPALS = c("rainbow", "topo.colors", "terrain.colors", "heat.colors", "tomo.col")  
pal = Gcols(plow=5, phi=0, N=100, pal=TPALS[3])
```

---

getmem

*Get Member*

---

**Description**

Get a member of a list

**Usage**

```
getmem(v, mem = 1)
```

**Arguments**

v	vector
mem	element in vector

**Details**

Used in conjunction with apply

**Value**

vector of members of a list

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>



**Examples**

```
z = list()
for(i in 1:10)
{
  z[[i]] = round(10*runif(10))
}

y = as.vector(unlist(lapply(z, getmem, 6)))
```

---

helpcolors

*Help on Personal Color Palettes*

---

**Description**

Give information on how to set up Personal Color Palettes

**Usage**

```
helpcolors()
```

**Value**

Side effects

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

palette

**Examples**

```
helpcolors()
```

---

HOZscale                      *add horizontal color scale*

---

### Description

Add horizontal color scale to existing plot.

### Usage

```
HOZscale(z, col, units="", SIDE=1, s1=.6, s2=0.95,
         format=1, digits=3, cex=1, cex.units=1)
```

### Arguments

z	image matrix
col	color palette
units	character string, units
SIDE	Side of the plot
s1	percent of margin for bottom
s2	percent of margin for top
format	Format: 1 for normal number, 2 for exponential notation
digits	Significant digits
cex	Character expansion for the numeric values.
cex.units	Character expansion for the units.

### Value

Graphical Side effect

### Author(s)

Jonathan M. Lees<jonathan.lees.edu>

### Examples

```
data(volcano)
image(volcano, col=terrain.colors(100))

HOZscale(volcano,terrain.colors(100) , units = "", SIDE = 1, s1 = 0.4, s2 = 0.95)

plot(1:10, 1:10, type='n')
j = c(runif(1, -10, 10) , runif(1, 20, 10000) )

### example showing scale above and below
HOZscale(j, terrain.colors(100),
```

```
units="hi", SIDE=3, s1=.4, s2=0.6, format=2, digits=2, cex.units = 1.2, cex=1.2)

j = c(runif(1, -10, 10)/1000 , runif(1, 1, 10) )

HOZscale(j, terrain.colors(100),
units="hi", SIDE=1, s1=.6, s2=0.8, format=2, digits=2, cex.units = 0.8)
```

---

ilocator

*Specialized Locator function*

---

### Description

Locator function with set parameters

### Usage

```
ilocator(N=1, COL=1, NUM=FALSE, YN=NULL, style=0)
```

### Arguments

N	number of points to locate
COL	color
NUM	number of points
YN	number of windows to span for lines
style	0,1,2 for differnt style of plotting vertical lines

### Details

if the window is divided into YN horizontal regions, style =2 will plot segments only within regions based on y-value of locator().

### Value

list:

x	x-locations
y	y-locations
n	number of points

### Author(s)

Jonathan M. Lees<jonathan.lees.edu>

### See Also

locator

**Examples**

```
plot(c(0,1), c(0,1), type='n')
for(i in 1:5) { abline(h=i/6) }

ilocator(N=3, COL = 1, NUM = 4, YN = 6, style = 2)
```

---

 itoxyz

*Vector Index to Matrix Index*


---

**Description**

Given I index get ix, iy, iz for three dimensional grids.

**Usage**

```
itoxyz(i, nx, ny, nz)
```

**Arguments**

i	index to long vector
nx	number of blocks in x axis
ny	number of blocks in y axis
nz	number of blocks in z axis (layers)

**Value**

ix	Index of X-array
iy	Index of Y-array
iz	Index of Z-array (layer)

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

xyztoi

**Examples**

```
itoxyz(24, 6, 6, 1)

kpos = itoxyz(2443:2500 , 20, 20, 13)
```

---

jpg

*png or pdf output*

---

### Description

Get file name and recreate plot on a png or pdf device. This program makes an attempt to keep the same size plot as viewed in the screen.

### Usage

```
jpg(file='tmp', width = 8, height = 8,P = NULL, bg = "white")  
jpgf(file='tmp', width = 8, height = 8,P = NULL)
```

### Arguments

file	png or pdf: will be added as a suffix, if needed
width	width, inches
height	height, inches
P	vector to fix the size, c(width, height)
bg	background color (default="transparent")

### Details

If  $P=c(10,12)$  is missing or NULL, program will attempt to use current plotting region via `par` to duplicated the size of the postscript device. Must close this device with `dev.off()` to finish. If either `w` or `h` are provided they will override the values in vector `P`.

If the standard suffix (png or pdf) are provided the file will be set. If these are omitted, they will be added to the given name according to the `local.file` function.

### Value

Graphical Side Effect

### Author(s)

Jonathan M. Lees<jonathan.lees.edu>

### See Also

`par`, `postscript`, `device`

**Examples**

```
## Not run:
jjj = local.file('hi', 'png')
x= rnorm(10)
y= rnorm(10)

plot(x,y)

print('resize the current plot')

jpng(jjj, width = 8, height = 8)
plot(x,y)
dev.off()

jpdf("HiThere.pdf", width = 8, height = 8 )
plot(x,y)
dev.off()

jpng("HiThere.png", width = 8, height = 8 , bg='red' )
plot(x,y)
dev.off()

## End(Not run)
```

---

jpostscript

*Postscript Output*


---

**Description**

Get file name and recreate plot on a postscript device. This program makes an attempt to keep the same size plot as viewed in the screen.

**Usage**

```
jpostscript(file=NULL, P=NULL, w=NULL, h=NULL)
```

**Arguments**

file	Postscript file name, eps will be added as a suffix
P	vector to fix the size, c(width, height)
w	width, inches
h	height, inches

**Details**

If  $P=c(10,12)$  is missing or NULL, program will attempt to use current plotting region via `par` to duplicated the size of the postscript device. Must close this device with `dev.off()` to finish. If either `w` or `h` are provided they will override the values in vector `P`.

**Value**

Graphical Side Effect

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

`par`, `postscript`, `device`

**Examples**

```
## Not run:
jjj = local.file('hi', 'eps')
x= rnorm(10)
y= rnorm(10)

plot(x,y)

print('resize the current plot')

jpostscript(jjj)
plot(x,y)
dev.off()

jpostscript("HiThere", P=c(7,7) )
plot(x,y)
dev.off()

jpostscript("HiThere", P=c(7,7), w=10 )
plot(x,y)
dev.off()

## End(Not run)
```

---

label.it	<i>Labels on Plots</i>
----------	------------------------

---

**Description**

Put Labels (A,B, C...) on corners of figures

**Usage**

```
label.it(a = "", corn = 1, ...)
```

**Arguments**

a	letters
corn	corner
...	graphical parameters passed from par

**Value**

Graphical Side effects

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```
par(mfrow=c(2,2))
for(i in 1:4)
{
plot(rnorm(5), rnorm(5))
label.it(letters[i],1)
}
```

---

local.file	<i>Get name for a Local file</i>
------------	----------------------------------

---

**Description**

Get a name for a local file for writing ascii files or postscript output. This code checks to see if file exists and if so it increments a counter in the name.

**Usage**

```
local.file(pref, suf)
```



**Arguments**

pref	prefix for file name
suf	suffix for file name

**Details**

File name is located in the current directory.

**Value**

character string for new file name

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
psfile = local.file("JML", "eps")
```

---

meshgrid	<i>Create a mesh grid like in Matlab</i>
----------	--

---

**Description**

Creates 2D matrices for accessing images and 2D matrices

**Usage**

```
meshgrid(a, b)
```

**Arguments**

a	x vector components
b	y vector components

**Details**

returns outer product of x-components and y-components for use as index arrays

**Value**

x	length(y) by length(x) matrix of x indicies
y	length(y) by length(x) matrix of y indicies

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```
meshgrid(1:5, 1:3)
```

---

 OPTREPLOT

*Replot Function for SELBUT*


---

**Description**

Replot Function for SELBUT

**Usage**

```
OPTREPLOT(opts , ncol=5, sel=1, HOZ=TRUE, TOP=TRUE,
  cols="white", scol="black", bcol="white" , tcol="black",
  slwd=1, blwd=3, main="", xlim=c(0,1), ylim=c(0,1),
  cex=1, mpct = 0.1, newplot=TRUE)
```

**Arguments**

opts	character list of options
ncol	number of columns
sel	vector of selected options
HOZ	logical, TRUE=plot horizontally
TOP	logical, TRUE=plot top-down
cols	colors
scol	select box color
bcol	default box color
tcol	box text color
slwd	select box line width
blwd	default box line width
main	character title
xlim	x-limits in plotting region (user coordinates)
ylim	y-limits in plotting region (user coordinates)
cex	character expansion for text in boxes
mpct	percentage margin to leave between option boxes
newplot	logical, TRUE=new plot

**Details**

Used internally in SELBUT as a replotting function

**Value**

list	
M	x,y matrix of grid
dx	delta x
dy	delta y
rx	range of x
ry	range of y

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

SELBUT, swig

**Examples**

```
STDLAB=c("DONE", "QUIT", "zoom.out", "zoom.in", "SELBUT", "FILT",  
"UNFILT", "PSEL", "SGRAM", "WLET", "SPEC", "XTR" )  
OPTREPLOT(STDLAB)
```

```
XMCOL = setXMCOL()  
YN = OPTREPLOT(XMCOL, cols =XMCOL, tcol=grey(.8) ,  
scol= "transparent", bcol= "transparent", mpct=0.05 )
```

```
YN = OPTREPLOT(XMCOL, cols =XMCOL, tcol=grey(.8) ,  
scol= "transparent", bcol= "black", mpct=0.05 )
```

---

`pastel.colors`      *pastel.colors*

---

**Description**

vector of pastel colors

**Usage**

```
pastel.colors(num, seed=0)
```

**Arguments**

<code>num</code>	number of colors
<code>seed</code>	random number seed

**Details**

The seed is a value given so that the same pastel colors can be extracted with each subsequent call to the code.

**Value**

vector of RGB hex colors

**Author(s)**

Jonathan M. Lees<[jonathan.lees@unc.edu](mailto:jonathan.lees@unc.edu)>

**See Also**

`rainbow`

**Examples**

```
pastel.colors(12)

pastel.colors(12, seed=1 )
```

---

pickcolors                    *Pick a SYSTEM color*

---

**Description**

Pick a SYSTEM color

**Usage**

```
pickcolors(COLLIST = colors(), BACK = "white")
```

**Arguments**

COLLIST	system colors
BACK	background for colors

**Value**

List of colors

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

syscolors

**Examples**

```
## Not run:
##### see named colors, excluding grey
SYSCOL = colors()
greys = grep('grey', SYSCOL)
grays = grep('gray', SYSCOL)

kolz = SYSCOL[-c(greys, grays) ]
pickcolors(COLLIST = kolz, BACK = "white")

### or just one type
SYSCOL = colors()
blues = SYSCOL[grep('blue', SYSCOL) ]
pickcolors(COLLIST = blues, BACK = "white")

## End(Not run)
```

rainbow.colors            *rainbow.colors*

---

**Description**

Color palette of n rainbow colors

**Usage**

```
rainbow.colors(n)
```

**Arguments**

n                      Nmber of colors desired

**Details**

rainbow.colors is set to match other color palette selections like topo.colors, terrain.colors

**Value**

Character vector of n colors from the default rainbow palette.

**Author(s)**

Jonathan M. Lees <jonathan.lees@unc.edu>

**See Also**

topo.colors, terrain.colors, palette

**Examples**

```
rainbow.colors(100)
```

---

RESCALE                      *Rescale a vector to fit in a certain range*

---

**Description**

Rescale a vector to fit in a certain range

**Usage**

```
RESCALE(x, nx1=0, nx2=1, minx=0, maxx=1)
```

**Arguments**

x	vector
nx1	new minimum
nx2	new maximum
minx	old min
maxx	old max

**Details**

Rescaling a vector, mostly used for graphics. If x does not vary, i.e. it is constant or minx and maxx are identical, the mean value of nx1 and nx2 is returned.

**Value**

Scale version of x vector is returned.

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
x = rnorm(10)
RESCALE(x, 3, 9, min(x), max(x) )
```

---

rowBUTTONS

*Geometry for the Really Poor Man's GUI*


---

**Description**

Create a set of buttons and associated geometry for RPGM

**Usage**

```
rowBUTTONS(labs, col = 6, pch = 4, cex=1, boxsize = -1)
```

**Arguments**

labs	Vector of labels for the buttons running across the top and bottom of the plot
col	Optional vector of colors for the buttons
pch	Optional vector of symbols to be plotted in the center of the buttons
cex	optional character expansion for text
boxsize	optional box size for the buttons, default=-1 where the size is adjusted for string size

**Details**

rowBUTTONS is called after the R graphic has been created so the geometry of the buttons can be set. Subsequent calls to whichbutt use the geometry to determine which button has been selected. Some of the parameters chosen here are controlled by par-like parameters.

**Value**

The function returns a list of buttons and the associated geometry.

N	Number of Buttons
labs	Names of the Buttons
x1	vector of left x-coordinates for the buttons
x2	vector of right x-coordinates for the buttons
y1	vector of top y-coordinates for the buttons
y2	vector of bottom y-coordinates for the buttons

**Note**

rowBUTTONS uses the current plotting parameters from par() to set the geometry. If the window is resized, rowBUTTONS should be reset to extract correct button position. In interactive mode this is done each time the plot is refreshed.

**Author(s)**

Jake Anderson and Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

whichbutt, par

**Examples**

```
##### create a plot
plot(c(0,1), c(0,1))
##### set the character vector of button labels
mybutts = c("DONE", "REFRESH", "rainbow", "topo", "terrain", "CONT",
"XSEC", "PS" )
##### set colors and plotting chars for buttons
colabs = rep(1, length(mybutts))
pchlabs = rep(0, length(mybutts))
##### create and set geometry for buttons:
buttons = rowBUTTONS(mybutts, col=colabs, pch=pchlabs)
```



---

see.pal	<i>plot a rectangular palette</i>
---------	-----------------------------------

---

**Description**

the function adds to an existing plot in the lower left corner

**Usage**

```
see.pal(col)
```

**Arguments**

col                    vector of colors

**Value**

Side Effects

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

see.pals

**Examples**

```
plot(c(0,1), c(0,1), type='n')
see.pal(rainbow(100))
```

---

SELOPT	<i>Select Options</i>
--------	-----------------------

---

**Description**

Select buttons interactively.

**Usage**

```
SELOPT(OPTS, onoff = -1, ncol=5, ocols = "white",
       cex=1, default="opt" )
```

**Arguments**

OPTS	character list of buttons
onoff	which buttons are active, onoff=-1 turns all buttons off, onoff=0 turns all buttons on, any other vector is an index vector to selected options
ncol	number of columns, default = 5
ocols	colors for plotting option boxes
cex	character expansion for text in boxes
default	default vector of options

**Details**

Used in swig. Options can be added, subtracted, deleted, or completely filled out based on interactive choice.

**Value**

character list of selected options

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

OPTREPLOT, chooser

**Examples**

```
## Not run:
STDLAB=c("DONE", "QUIT", "zoom.out", "zoom.in", "SELOPT",
"FILT", "UNFILT",
"PSEL", "SGRAM", "WLET", "SPEC", "XTR" )
onoff = rep(0, length(STDLAB))
onoff[1:5] = 1
SELOPT(STDLAB, onoff=onoff)

### second option for selecting colors
###dev.new(width=12, height=12)

scol = SELOPT(colors(), onoff=-1, ncol=15, ocols =colors(), cex=.6 )

### old program
SHOWPAL(scol, NAME=TRUE)

### show the options chosen from top to bottom
OPTREPLOT(scol, cols=scol, scol="green", bcol="blue", slwd=15 )
```

```
## End(Not run)
```

---

sepia.colors

*Sepia Color Palette*

---

## Description

Sepia Color Palette

## Usage

```
sepia.colors(n, k = 1)  
hcl.colors(n, k = 260)
```

## Arguments

n	Number of colors
k	Sepia starting color, hcl ending number

## Details

There are two version of sepia in the code, each has a slightly different sepia end member.

## Value

vector of Octal color codes

## Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

## See Also

tomo.colors, pastel.colors, syscolors, helpcolors

## Examples

```
scol = sepia.colors(100)  
SHOWPAL(scol)  
see.pal(scol)
```

setXMCOL                      *Set up color map from Geotouch*

---

**Description**

Uses colors predefined in geotouch

**Usage**

```
setXMCOL()
```

**Value**

Vector of named colors

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```
XMCOL=setXMCOL()
```

---

shade.col                      *Shaded Color Palette*

---

**Description**

Create a color palette with two end member colors

**Usage**

```
shade.col(n, acol = c(1, 0, 0), bcol = c(1, 1, 1))
```

**Arguments**

n	number of desired colors
acol	rgb, starting color
bcol	rgb, ending color

**Details**

Linear interpolation from color1 to color 2.

**Value**

color vector

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

rainbow, tomo.col

**Examples**

```
## color palette from red to white
shade.col(100, acol = c(1, 0, 0), bcol = c(1, 1, 1))
```

---

SHOWPAL

*Show a palette of colors as a bar*

---

**Description**

Show a palette of colors as a bar

**Usage**

```
SHOWPAL( COLLIST , NAME = FALSE, NUM=FALSE, ncol = 5, BACK="transparent")
```

**Arguments**

COLLIST	vector of colors
NAME	name of palette
NUM	logical, TRUE=show index number
ncol	number of colors
BACK	Background color, default=NULL

**Value**

Graphical Side Effects

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

see.pals, help.pal , plotpal , helpcolors

## Examples

```
##### make a large screen for a lot of colors
### dev.new(width=12, height=12)

SHOWPAL(colors(), ncol=15, NAME=FALSE)

gcol = setXMCOL()

SHOWPAL(gcol, ncol=10, NAME=TRUE)

#### show index:
SHOWPAL(gcol, ncol=10, NAME=TRUE, NUM=TRUE)

p1 = c("grey", "lightblue1", "pink", "darkseagreen2", "gold1",
      "chartreuse1", "aquamarine", "plum1", "goldenrod", "maroon1",
      "deepskyblue", "palegreen2", "salmon")

SHOWPAL(p1, NAME=TRUE, NUM=TRUE)

SYSCOL = pastel.colors(100)
SHOWPAL(SYSCOL, ncol=10)

SYSCOL = sepia.colors(100)
SHOWPAL(SYSCOL, ncol=10)

SYSCOL = hcl(h=seq(from=0, to=260, length=100) )
SHOWPAL(SYSCOL, ncol=10)
```

---

slideshow

*SlideShow*

---

## Description

MAke a slide show similar to Powerpoint presentations

## Usage

```
slideshow(P = c("hi", "there", "sugar pie"),
dy = 0.2, EX = 0.1, ht = 3, font = 2, anim = FALSE)
```

**Arguments**

P	vector of character strings to display
dy	vertical spacing, percentage
EX	horizontal offset, percentage
ht	Character expansion, see par
font	Font choice, see par
anim	logical, Animation, TRUE=means animate the input line-by-line

**Details**

The function is meant to be used in presentations when R is running a script and text needs to be displayed to explain the talk. The animation is controlled by clicking on the screen using locator(1) function.

**Value**

Side effects

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```
Ptext1 = c("New Package: Rquake", "Earthquake Location",
           "Inverse Theory",
           "Graphics",
           "Statistical Analysis" )

slideshow(Ptext1, ht=3, anim=FALSE )
```

---

textrect

*Text labels with border*

---

**Description**

Plot Text labels with border and background color

**Usage**

```
textrect(x, y, lab, textcol = "black", col = "white",
         border = "black", off = 0.06, brd = 0.06, pos = 1, log="" ,
         add=TRUE, ...)
```

**Arguments**

x	x-location, user coordinates
y	y-location, user coordinates
lab	character for label
textcol	color for labels
col	color for background
border	color for border, NA=do not plot
off	Offset from point, inches, default=0.06
brd	Border around text, inches, default=0.06
pos	numeric, position=one of (0.0, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5), as in the normal text call with pos=1,2,3,4, however, here I allow half way between points. 0 indicates no offset and label is placed centered on the point.
log	character, as in plot
add	add to existing plot (FALSE returns plotting rectangles)
...	additional parameters from par, used for font, cex, etc...

**Details**

textrect plots a label on an existing plot at the location designated. The text is surrounded by a rectangular box with color inside and a border. The box can be placed around the designated point at 9 positions. Positions 1,2,3,4 are the same as text parameter pos. Position 0 is centered, i.e. no offset. Positions, 1.5, 2.5, 3.5, 4.5 are at an angle 45 degrees clockwise from the integer values.

**Value**

graphical side effects.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```

thepos = c(0, seq(from=1, to=4.5, by=.5))
lab="the string"

x = 1:9
y = 1:9
plot(x,y, asp=1)
for(i in 1:length(thepos))
{
  textrect(x[i], y[i], lab, col=i , border='green' ,
  textcol="gold", off=.06, brd=.06 , pos=thepos[i], font=1, cex=.8 )
}

```



```

x = runif(10)
y = runif(10)
lab = floor( 1000*runif(10) )
i=sample(thepos, 10, replace = TRUE)
col = sample(rainbow(100) , 10, replace = TRUE)

plot(x,y, asp=1)
textrect(x, y, lab, pos=i , textcol="black", col=col)

```

---

VVwheel

*Make a color rectangle (wheel)*


---

### Description

Make a color rectangle (wheel)

### Usage

```
VVwheel(BIGMESH = NULL, v = 1)
```

### Arguments

BIGMESH	color mesh
v	v, from hsv color scheme

### Value

M	meshgrid: x x - location y y - location
ARE	Radii
pANG	angle
dx	delta x
dy	delta y
RY	range x
RX	range y

### Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

hsv, VVwheel, wheelrgb

**Examples**

```
## Not run:  
BIGMESH = VVwheel( v=1)  
  
## End(Not run)
```

---

wheelrgb

*Plot a large color rectangle for color selection*

---

**Description**

Plot a large color rectangle for color selection

**Usage**

```
wheelrgb(wloc, v, RY)
```

**Arguments**

wloc	output of locator
v	v, from hsv color scheme
RY	coordinates of meshgrid, output of VVwheel

**Value**

vector of colors

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

colwheel, VVwheel

---

whichbutt	<i>Determines which button was selected in RPGM</i>
-----------	---

---

**Description**

Function to determine which button of the RPGM was selected during a graphics session.

**Usage**

```
whichbutt(v, buttons)
```

**Arguments**

v	list of x,y coordinates obtained from the locator() function
buttons	list of buttons set by the function rowBUTTONS

**Details**

whichbutt uses the geometry determined by rowButtons and a list of locator() points to return the buttons clicked on or, if none, 0.

**Value**

Returns a vector of indexes to buttons selected by the user. Buttons are numbered 1-N so if a click is not on a button, zero is returned.

**Note**

This function can be used to get interaction with predefined buttons and non-button clicks using locator().

**Author(s)**

Jonathan M. Lees <jonathan.lees@unc.edu>

**See Also**

rowBUTTONS, locator

**Examples**

```
##### initial plot
plot(c(0,1), c(0,1))
##### set buttons
mybutts = c("DONE", "REFRESH", "rainbow", "topo", "terrain", "CONT",
"XSEC", "PS" )
colabs = rep(1, length=length(mybutts))
pchlabs = rep(0, length(mybutts))
```

```
##### set button geometry
buttons = rowBUTTONS(mybutts, col=colabs, pch=pchlabs)
##### user clicks on plot. When locator finishes, whichbutt
##### determines which buttons were selected and returns the vector
L = locator()

K = whichbutt(L, buttons)
print(K)
```

---

writeCOMMENT

*write Code Comments*


---

### Description

Create a print out of comments for insertion in computer code. Used for separating important blocks of code with helpful, easy to find comments.

### Usage

```
writeCOMMENT(temp, space = " ", letspace = "", MSUB = "0", prefix = "", suffix = "")
```

### Arguments

temp	text string
space	space between words
letspace	space between letters
MSUB	text, substitute character, if this is "ALL", then each letter is substituted. default=NULL
prefix	prefix before the letters
suffix	suffix after the letters

### Details

This is a function used for creating comments in computer code. Letters are a fixed height of 7 lines

### Value

List	26 letters
------	------------

### Note

Code dumps to the screen, then you must paste in code. If sent in an email, spaces are not preserved. The letters are stored in the routine, these can be changed, but the constant (7 lines) common height should be preserved. Each letter should be one block.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```

writeCOMMENT("GO TARHEELS", space="      ", letspace = "", MSUB="ALL", prefix="/*" , suffix="*/" )
  writeCOMMENT("START", space="      ", letspace = "", MSUB="ALL", prefix="#####" )
writeCOMMENT("J M lees", space="      ", letspace = "", MSUB="0" )

writeCOMMENT("J. M. Lees", space="      ", letspace = "", MSUB="0" )
writeCOMMENT("J. M. Lees", space="      ", letspace = "", MSUB="." )

writeCOMMENT("J. M. Lees", space="      ", letspace = "" )
writeCOMMENT("J. M. Lees", space="-----", letspace = "" )

writeCOMMENT("J. M. Lees", space="      ", letspace = "", MSUB="ALL" )
writeCOMMENT("J_M_Lees", space="      ", letspace = "", MSUB="ALL" )

writeCOMMENT("abcdefghi")
writeCOMMENT("jklmnop")

writeCOMMENT("qrstuvwxyz")
writeCOMMENT("1234567890")
writeCOMMENT("WHY?!.-+="_)
writeCOMMENT("2+2=4")
writeCOMMENT("e*exp(pi*i)=-1")

```

---

XSECDEM

*Cross Sections Using RPMG*


---

**Description**

This function Takes a Digital Elevation Map (or any surface) and illustrates how to take interactive cross sections with RPMG through the surface.

**Usage**

```
XSECDEM(Data, labs, demo=FALSE)
```

**Arguments**

Data	Structure with x, y, z components, typical of contoured surfaces or digital images
labs	Vector of labels for Buttons used in the RPMG
demo	Argument used to turn off interactive part. Default is FALSE, but for package construction is set to TRUE so no interaction is required.

**Details**

XSECDEM is an example stub illustrating the use of RPMG. The idea is to set up a while() loop that uses input from the locator() function to execute or analyze data depending on user defined buttons. Actions are executed when the button clicked matches the list of names provided by the user.

**Value**

No return values

**Note**

This code is designed as an example of how to set up a Really Poor Man's GUI. The demo argument is supplied so that this code will run without user input, as when creating a checks for package construction.

**Author(s)**

Jonathan M. Lees <jonathan.lees@unc.edu>

**See Also**

whichbutt, rowBUTTONS

**Examples**

```
data(volcano)
attr(volcano, 'dx') =10
attr(volcano, 'dy') =10
mybutts = c("DONE", "REFRESH", "rainbow", "topo", "terrain", "CONT",
"XSEC", "PS" )
### in the following change demo=FALSE to get interactive behavior
XSECDEM(volcano, mybutts, demo=TRUE)
```

---

`xyztoi`*Matrix Index to Vector index*

---

**Description**

Given ix, iy, iz index get I.

**Usage**

```
xyztoi(ix, iy, iz, nx, ny, nz)
```

**Arguments**

<code>ix</code>	index to col vector
<code>iy</code>	index to row vector
<code>iz</code>	index to (depth) layer vector
<code>nx</code>	number of blocks in x axis
<code>ny</code>	number of blocks in y axis
<code>nz</code>	number of blocks in z axis (layers)

**Value**

<code>i</code>	Index of matrix
----------------	-----------------

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

`itoxyz`

**Examples**

```
k = itoxyz(24, 6, 6, 1)
xyztoi(k$ix, k$iy, k$iz, 6, 6, 1)
```

```
nx = 20
ny = 20
nz = 40
```

```
k = itoxyz(2440, nx, ny, nz)
xyztoi(k$ix, k$iy, k$iz, nx, ny, nz )
```

---

`ymarginfo`*Get information on Y-margin for plotting*

---

**Description**

Get information on Y-margin for plotting

**Usage**

```
ymarginfo(SIDE = 1, s1 = 0.1, s2 = 0.8)
```

**Arguments**

<code>SIDE</code>	plotting side 1,2,3,4
<code>s1</code>	lower percent of margin to return
<code>s2</code>	upper percent of margin to return

**Details**

Function uses `par` to help determine how to plot objects in the margins.

**Value**

vector `c(a, b)` giving coordinates in margin worth plotting.

**Author(s)**

Jonathan M. Lees<[jonathan.lees.edu](mailto:jonathan.lees.edu)>

**See Also**

`par`

**Examples**

```
plot(c(0,1), c(0,1), type='n')
s1=0.4
s2=0.95
ym = ymarginfo(SIDE=1, s1=s1, s2=s2)
```



# Index

## \*Topic **aplot**

circle, 10  
label.it, 24  
rainbow.colors, 30  
see.pal, 33

## \*Topic **hplot**

HOZscale, 18  
SHOWPAL, 37  
VVwheel, 41

## \*Topic **iplot**

aGETXprofile, 4  
chooser, 8  
ilocator, 19  
rowBUTTONS, 31  
whichbutt, 43  
XSECDM, 45

## \*Topic **misc**

breakline.index, 5  
butdoc, 7  
colwheel, 11  
cprint, 12  
endSCALE, 13  
fmod, 14  
Gcols, 15  
getmem, 16  
helpcolors, 17  
itoxyz, 20  
jpg, 21  
jpostscript, 22  
local.file, 24  
meshgrid, 25  
OPTREPLOT, 26  
pastel.colors, 28  
pickcolors, 29  
RESCALE, 30  
SELOPT, 33  
sepia.colors, 35  
setXMCOL, 36  
shade.col, 36

slideshow, 38  
textrect, 39  
VVwheel, 41  
wheelrgb, 42  
writeCOMMENT, 44  
xyztoi, 47  
ymarginfo, 48

## \*Topic **package**

RPMG-package, 2

aGETXprofile, 4

breakline.index, 5  
butdoc, 7

chooser, 8  
circle, 10  
colwheel, 11  
cprint, 12

endSCALE, 13

fmod, 14

Gcols, 15  
getmem, 16

hcl.colors (sepia.colors), 35  
helpcolors, 17  
HOZscale, 18

ilocator, 19  
itoxyz, 20

jpdf (jpg), 21  
jpg, 21  
jpostscript, 22

label.it, 24  
local.file, 24

meshgrid, 25

OPTREPLOT, [26](#)

pastel.colors, [28](#)  
pickcolors, [29](#)

rainbow.colors, [30](#)  
RESCALE, [30](#)  
rowBUTTONS, [31](#)  
RPMG (RPMG-package), [2](#)  
RPMG-package, [2](#)

see.pal, [33](#)  
SELOPT, [33](#)  
sepia.colors, [35](#)  
setXMCOL, [36](#)  
shade.col, [36](#)  
SHOWPAL, [37](#)  
showtopopal (SHOWPAL), [37](#)  
slideshow, [38](#)

textrect, [39](#)

VVwheel, [41](#)

wheelrgb, [42](#)  
whichbutt, [43](#)  
writeCOMMENT, [44](#)  
writeCOMMENT2 (writeCOMMENT), [44](#)

XSECDEM, [45](#)  
xyztoi, [47](#)

ymarginfo, [48](#)