

# Package ‘DAAGbio’

January 20, 2025

**Version** 0.63-4

**Date** 2023-08-19

**Title** Data Sets and Functions, for Demonstrations with Expression  
Arrays and Gene Sequences

**Author** John Maindonald

**Maintainer** John Maindonald <jhmaindonald@gmail.com>

**Description** Data sets and functions, for the display of gene expression array (microarray) data, and for demonstrations with such data.

**LazyLoad** true

**LazyData** true

**Depends** R (>= 3.0.0), limma (>= 2.9.15)

**Suggests** DAAG, locfit, knitr

**ZipData** no

**License** GPL (>= 2)

**VignetteBuilder** knitr

**URL** <https://github.com/jhmaindonald/DAAGbio/>

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2023-08-20 23:02:36 UTC

## Contents

coralRG . . . . .	2
coralTargets . . . . .	2
DEnma . . . . .	3
imgplot . . . . .	3
plantStressCounts . . . . .	8
plotprintseq . . . . .	9
primateDNA . . . . .	10
xplot . . . . .	11
<b>Index</b>	<b>13</b>

coralRG

*Spotted microarray red and green foreground and background values*

---

**Description**

Unnormalised red and green values, and corresponding background values. Further information is in the data frame coralTargets

**Usage**

```
data(coralRG)
```

**Format**

The format is: Formal class 'RGList' [package "limma"] Can be accessed as a list, with named elements "R" , "G" , "Rb" , "Gb" , "targets" , "source" , "genes" and "printer"

**Source**

Lauretto Grasso and Eldon Ball, Molecular Genetics and Evolution Group, Research School of Biological Sciences, Australian National University.

**Examples**

```
data(coralRG)
```

---

coralTargets

*Targets file to accompany spotted expression array data*

---

**Description**

Targets file, in the form expected by limma, to accompany the expression array data in coralRg

**Usage**

```
data(coralTargets)
```

**Format**

A data frame with 6 observations on the following 4 variables.

SlideNumber a character vector

FileName Names of files that hold spotted array data

Cy3 Treatment assigned to Cy3 ("red")

Cy5 Treatment assigned to Cy5 ("green")

**Examples**

```
data(coralTargets)
## maybe str(coralTargets) ; plot(coralTargets) ...
```

---

DEnma	<i>Spotted microarray M and A values; differentially expressed controls</i>
-------	---

---

**Description**

Values, derived from the data in coralRG, are the subset of the M and A values, after normalisation within and between arrays, for the differentially expressed controls

**Usage**

```
data(DEnma)
```

**Format**

The format is: Formal class 'MAList' [package "limma"] Can be accessed as a list, with named elements "targets", "source", "genes", "printer", "M" and "A"

**Source**

Centre for the Molecular Genomics of Genetic Development, ANU

**Examples**

```
data(DEnma)
```

---

imgplot	<i>Image plot of spotted expression array data</i>
---------	--

---

**Description**

Creates an image of graduated colors that represent the values of a statistic for each spot on a spotted microarray. By default, the only the 5 shown. The initial version was based on plot.spatial in the sma package.

**Usage**

```
imgplot(z = DAAGbio::coralRG$R[, 1], layout = DAAGbio::coralRG$printer, crit1 = 0.05,
crit2 = crit1, key.side=2,
lohi.colors = c("#9E0142", "#D53E4F", "#F46D43", "#FDAE61", "#ABDDA4",
"#66C2A5", "#3288BD", "#5E4FA2"), nacolor = "#FFFF00",
boxplot.side = 1, split = "quantiles")
```

**Arguments**

z	values to be plotted
layout	layout of spots, in the order (rows of grids, columns of grids, rows of spots in a grid, columns in a grid)
crit1	Choose the lower threshold to include this proportion at the high end
crit2	Choose the upper threshold to include this proportion of values at the low end
key.side	Side on which the color key should appear
lohi.colors	Graduated sequence of colors
nacolor	Use this color for NAs
boxplot.side	Show boxplot on this side of figure region
split	Specify "intervals" or "quantiles", as required

**Value**

A plot is created on the current graphics device

**Author(s)**

J. H. Maindonald

**Examples**

```
## The function is currently defined as
function (z=DAAGbio::coralRG$R[,1], layout=cDAAGbio::oralRG$printer, crit1 = 0.05,
         crit2 = crit1, key.side=2,
         lohi.colors=c("#9E0142", "#D53E4F", "#F46D43", "#FDAE61",
                       "#ABDDA4", "#66C2A5", "#3288BD", "#5E4FA2"),
         nacolor="#FFFF00", boxplot.side=1, split="quantiles")
{
  "block2matrix" <-
  function(z, sr=3, sc=2, gr=2, gc=2){
    ## Assumes that values in the vector z are in row major
    ## order within blocks of dimension sr x sc, with blocks
    ## in row major order within a gr x gc array of grids.
    ## Elements in the vector that is returned are in row
    ## major order wrt the sr*gr x sc*gc matrix of values on
    ## the slide. (It is given the dimensions of a matrix.)
    xy <- array(z, dim=c(sc, sr, gc, gr))
    xy <- aperm(xy, c(1,3,2,4))
    dim(xy) <- c(sc*gc, gr*sr)
    xy}
  quantile.na <- function (z, ...)
  {
    tmp <- !(is.na(z) | is.infinite(z))
    quantile(z[tmp], ...)
  }
  length.na <- function (z, ...)
  {
```

```

    tmp <- !(is.na(z) | is.infinite(z))
    length(z[tmp], ...)
  }
  if(is.matrix(z))warning("z is a matrix, You probably want a column vector")
bplot <- function(z, boxplot.side=1){
  xrange <- range(z,na.rm=TRUE)
  iqr <- diff(quantile(xrange, c(.25,.75)))
  bwex <- diff(xrange)/(3*iqr)
  xhi <- max(z,na.rm=TRUE)
  xusr <- par()$usr[c(1:2)]
  xpos=pretty(z[!is.na(z)], n=5)
  z <- xusr[1]+(z-xrange[1])*diff(xusr)/diff(xrange)
  newpos <- xusr[1]+(xpos-xrange[1])*diff(xusr)/diff(xrange)
  par(xpd=TRUE)
  atvert <- switch(boxplot.side, par()$usr[3]-par()$cxy[2]*0.8,
    "", par()$usr[4]+par()$cxy[2]*0.8, "")
  if(atvert!=""){
    boxplot(z, at=atvert, boxwex=bwex, add=TRUE, horizontal=TRUE, xaxt="n")
    axis(side=boxplot.side, line=1.5,
      at=newpos, labels=xpos, cex.axis=0.75, mgp=c(2, 0.5, 0))
  }
  par(xpd=FALSE)
}
}
if (crit1 >= 1)
  crit1 <- crit1/(length.na(z))
if (crit2 >= 1)
  crit2 <- crit2/(length.na(z))
tmpind <- (z > quantile.na(z, probs = 1 - crit2)) | (z <
  quantile.na(z, probs = crit1))

n <- prod(unlist(layout))
n.all <- length(z)
n.na <- sum(is.na(z))
nhalf <- length(lohi.colors)%/2
n2 <- 2*nhalf
n.one <- length(lohi.colors)
plo <- crit1*(0:nhalf)/nhalf
phi <- 1-crit2*(nhalf:0)/nhalf
quiles1 <- quantile.na(z, plo)
quiles2 <- quantile.na(z, phi)
if(split=="intervals"){
  quiles1[2:nhalf] <- quiles1[1]+(quiles1[nhalf+1]-quiles1[1])*
    (1:(nhalf-1))/nhalf
  quiles2[2:nhalf] <- quiles2[1]-(quiles2[nhalf+1]-quiles2[1])*
    ((nhalf-1):1)/nhalf
  plo[-1] <- sapply(quiles1[-1],
    function(x, z)sum(z<=x, na.rm=TRUE)/length.na(z), z=z)
  phi[-1] <- sapply(quiles2[-1],
    function(x, z)sum(z<=x, na.rm=TRUE)/length.na(z), z=z)
}

if(crit1+crit2<1){
  quiles <- c(quiles1,quiles2)
  frac <- c(plo, phi)

```

```

    colpal <- c(lohi.colors[1:nhalf], "#FFFFFF",
               lohi.colors[(n.one-nhalf+1):(n.one)])
    midbreak <- TRUE
  }
  else {colpal <- lohi.colors
        midbreak <- FALSE
        quiles <- quantile.na(z, (0:n.one)/n.one)
        frac <- c(plo, phi[-1])
      }
  dups <- duplicated(quiles)
  if(any(dups)){
    cats <- seq(along=quiles[-1])
    filledcats <- cats[!dups]
    cutcats <- as.integer(cut(z, quiles[!dups], include.lowest=TRUE))
    fullm <- filledcats[cutcats]}
  else fullm <- as.integer(cut(z, quiles, include.lowest=TRUE))
  n.one <- length(colpal)
  nrects <- length(quiles)
  if(any(is.na(z))){
    nacat <- TRUE
    fullm[is.na(fullm)] <- max(unique(fullm[!is.na(fullm)]))+1
    colpal <- c(colpal, nacolor)
  }
  else nacat <- FALSE
  if ((length(as.vector(z)) != n) & (!is.null(names(z)))) {
    y <- fullm[tmpind]
    fullm <- rep(NA, n)
    fullm[as.integer(names(y))] <- y
  }
  else fullm[!tmpind] <- NA
  if ((length(as.vector(z)) != n) & (is.null(names(z)))) {
    stop(paste("Error: Length of vector is different from total number\n",
              "of spots and vector has no row.name.\n"))
  }
#####
gc <- layout$ngrid.c
gr <- layout$ngrid.r
sc <- layout$nspot.c
sr <- layout$nspot.r
full <- block2matrix(fullm, sr, sc, gr, gc)
image(1:ncol(full), 1:nrow(full), t(full), axes = FALSE,
      xlab = "", ylab = "", col=colpal)
box()
abline(v = ((gr - 1):1) * (sr) + 0.5)
abline(h = (1:(gc - 1)) * (sc) + 0.5)
#####
if(boxplot.side%in%c(1,3))bplot(z, boxplot.side=boxplot.side)
if(key.side%in%c(2,4)){
  chw <- par()$cxy[1]
  barwid <- 0.75*chw
  if(key.side==2){
    x0 <- par()$usr[1]-chw-barwid
    xcutpos <- x0 - 0.4*chw
  }
}

```

```

    xquilepos <- x0+barwid+0.55*chw
    srt <- 90
  }
  else {
    x0 <- par()$usr[2]+chw
    xcutpos <- x0 + barwid + 0.4*chw
    xquilepos <- x0-0.4*chw
    srt <- -90
  }
  yvals2 <- seq(from=par()$usr[3], to=par()$usr[4],
               length=n2+midbreak+2*nacat+1)[-(n2+midbreak+2*nacat+1)]
  eps2 <- diff(yvals2[1:2])

  if(nacat){
    nlast <- length(yvals2)
    nclast <- length(colpal)
    rect(x0, yvals2[nlast], x0+barwid, yvals2[nlast]+eps2,
         col=colpal[nclast], xpd=TRUE)
    text(x0+0.5*barwid, yvals2[nlast]+0.5*eps2, "NA",
         xpd=TRUE, srt=srt)
    yvals2 <- yvals2[-((nlast-1):nlast)]
    colpal <- colpal[-nclast]
  }
  if(!midbreak){
    rect(x0, yvals2, x0+barwid, yvals2+eps2,
         col=colpal, xpd=TRUE)
    text(xcutpos, c(yvals2[1],yvals2+eps2),
         paste(signif(quiles,3)), srt=srt, xpd=TRUE, cex=0.8)
    text(xquilepos, yvals2[1], "(%)", srt=srt, xpd=TRUE, cex=0.65)
    frac <- frac[-c(1, length(frac))]
    text(xquilepos, yvals2[-1],
         paste("(",round(frac*100,2),")",sep=""),
         srt=srt, xpd=TRUE, cex=0.65)
    text(xquilepos, yvals2[length(yvals2)]+eps2, "(100%)", srt=srt,
         xpd=TRUE, cex=0.65)
  }
  else {rect(x0, yvals2[1:nhalf], x0+barwid, yvals2[1:nhalf]+eps2,
            col=colpal[1:nhalf], xpd=TRUE)
        rect(x0, yvals2[(nhalf+2):(2*nhalf+1)], x0+barwid,
            yvals2[(nhalf+2):(2*nhalf+1)]+eps2,
            col=colpal[(nhalf+2):(2*nhalf+1)], xpd=TRUE)
        text(xcutpos, yvals2[1:(nhalf+1)],
            paste(signif(quiles1,3)), srt=srt, xpd=TRUE, cex=0.8)
        text(xquilepos, yvals2[2:(nhalf+1)],
            paste("(",round(plo[-1]*100,2),")",sep=""), srt=srt,
            xpd=TRUE, cex=0.65)
        text(xquilepos, yvals2[1], "(%)", srt=srt, xpd=TRUE, cex=0.65)
        text(xcutpos,
            c(yvals2[(nhalf+2):(2*nhalf+1)], yvals2[2*nhalf+1]+eps2),
            paste(signif(quiles2,3)), srt=srt, xpd=TRUE, cex=0.8)
        text(xquilepos, yvals2[(nhalf+2):(2*nhalf+1)],
            paste("(",round(phi[-length(phi)]*100,2),")",sep=""),

```

```

        srt=srt, xpd=TRUE, cex=0.65)
    text(xquilepos, yvals2[2*nhalf+1]+eps2, "(100%)", srt=srt,
         xpd=TRUE, cex=0.65)
  }
}
invisible()
}

```

---

plantStressCounts      *Matrix holding mRNA counts*

---

### Description

Three treatments (3 samples each) were applied to Arabidopsis plants. RNA-Seq technology was used to determine messenger RNA (mRNA) counts. These were processed to remove counts for sequences that could not be identified as corresponding to a gene.

### Usage

```
data("plantStressCounts")
```

### Format

The matrix plantStressCounts has 28775 rows, and 9 columns. Rows have the nondescript names "Gene1" "Gene2" "Gene3" "Gene4" ... . Columns are named "CTL1", "CTL2", "CTL3", "Light1", "Light2", "Light3", "Drought1", "Drought2", "Drought3"

### Details

The treatments were:

**Control** Plants were grown under normal light and watering conditions

**Light stress** One hour of continuous exposure to light at ten times the level that the plants are normally grown under

**Drought stress** Nine days without water, causing wilting of the leaves

The interest is in how light and drought stress affect gene expression to produce proteins.

### Source

Data are from Peter Crisp, obtained as part of his PhD work in the ARC Centre of Excellence in Plant Energy Biology at Australian National University.

### Examples

```

data(plantStressCounts)
## maybe str(plantStressCounts) ; plot(plantStressCounts) ...

```



---

plotprintseq

*Sequence of movements of spotted microarray printhead*


---

**Description**

Shows the sequence of movements of a spotted microarray printhead, when a slide is printed.

**Usage**

```
plotprintseq(ngrid.r = 4, ngrid.c = 4, nspot.r = 16, nspot.c = 12,
  gridorder = expand.grid(row = 1:ngrid.c, col = 1:ngrid.r),
  spotorder = list(x = nspot.r:1, y = nspot.c:1), rowmajor = FALSE, eps =
  1, delay1 = 100,
  delay2 = 2000)
```

**Arguments**

ngrid.r	Number of rows of grids
ngrid.c	Number of columns of grids
nspot.r	Number of rows of spots in a grid
nspot.c	Number of columns of spots in a grid
gridorder	A data frame whose rows specify grids, in order of printing
spotorder	A list, specifying the order across rows and up or down each column in a grid
rowmajor	Order of printing of spots within grids.
eps	Distance between grids
delay1	Delay in shifting by one spot
delay2	Delay in shifting to new column or new row

**Examples**

```
plotprintseq()

## The function is currently defined as
function(ngrid.r=4, ngrid.c=4,
  nspot.r=16, nspot.c=12,
  gridorder=expand.grid(row=1:ngrid.c, col=1:ngrid.r),
  spotorder=list(x=nspot.r:1, y=nspot.c:1),
  rowmajor=FALSE, eps=1, delay1=100, delay2=2000){
  oldpar <- par(mar=par())$mar-c(0,0,2.5,0)
  on.exit(par(oldpar))
  plotpoints <- function(i, j, delay1=5000, delay2=10000){
    points(i+xy$x, j+xy$y, pch=15,
      cex=0.5, col="cyan")
    x <- 0
    for(k in 1:delay2)x <- x+1
    points(i+xy$x, j+xy$y, pch=15,
```

```

        cex=0.85, col="grey60")
x <- 0
for(k in 1:delay1)x <- x+1
}

xy <- gridorder-1
names(xy) <- c("x","y")
xy$x <- xy$x*(nspot.c+eps)
xy$y <- xy$y*(nspot.r+eps)
plot(c(1, ngrid.c*(nspot.c+eps)),
      c(1, ngrid.r*(nspot.r+eps)),
      type="n",xlab="",ylab="",axes=FALSE)
mtext(side=1, line=1,
       paste("Grid layout: #rows of Grids =", ngrid.r,
             " #columns of Grids =", ngrid.c))
mtext(side=1, line=2.5,
       paste("In each grid: #rows of Spots =", nspot.r,
             " #columns of Spots =", nspot.c))
if (rowmajor)
for(j in spotorder$x) for(i in spotorder$y)
plotpoints(i,j, delay1=delay1, delay2=delay2)
else
for(i in spotorder$y) for(j in spotorder$x)
plotpoints(i,j, delay1=delay1, delay2=delay2)
}

```

---

primateDNA

*Mitochondrial DNA sequence data from 14 primates*


---

### Description

Bases at 232 mitochondrial locations (not continuous), for each of 14 primates.

### Usage

```
data(primateDNA)
```

### Format

A matrix of 14 rows (primate species) by 232 locations.

### Source

Data, originally from Masami Hasegawa, are from <http://evolution.genetics.washington.edu/book/primates.dna>

### References

Felsenstein, J. 2003. *Inferring Phylogenies*. Sinauer.

**Examples**

```

data(primateDNA)
## Not run:
library(ape)
primates.dist <- dist.dna(as.DNAbin(primateDNA), model = "K80")
primates.cmd <- cmdscale(primates.dist)
left <- primates.cmd[,1] <- apply(primates.cmd, 1, mean)
plot(primates.cmd)
text(primates.cmd, rownames(primates.cmd), pos=left*2+2)

## End(Not run)

```

xplot

*Repeat specified plot across multiple columns of a matrix***Description**

This is designed to repeat a plot, usually an image plot, across multiple columns of a matrix of gene expression values. A boxplot that shows the distribution of values appears below each panel.

**Usage**

```

xplot(data = DAAGbio::coralRG$R, images = 1:6, layout = DAAGbio::coralRG$printer, mfrow =
c(3, 2),
FUN = imgplot, device = NULL, title = NULL, width = 7.5, height = 10,
paneltitles = c("1:R/G", "2:G/R", "3:R/G", "4:G/R", "5:R/G", "6:G/R"),
paneltitles.line = 0.5,
mar = c(3.6, 3.6, 1.6, 0.6), oma = c(0.6, 0.6, 1.6, 0.6), file = NULL)

```

**Arguments**

data	matrix of expression array values
images	columns of matrix for which plots are required
layout	layout of spots, in the order (rows of grids, columns of grids, rows of spots in a grid, columns in a grid)
mfrow	row by column layout of plots on a page
FUN	imgplot, or imageplot from limma
device	If NULL, plot appears on the monitor. Other possibilities include pdf, postscript, png, jpeg and bitmap
title	A title for the page of graphs
width	width of plot (in)
height	height of plot (in)
paneltitles	character vector of titles for individual panels

```

paneltitles.line      height (lines) at which panel title are to appear above the upper margin of each
                      panel
mar                   Setting for par$mar
oma                   Setting for par$mar
file                   Optional file name, if output is to a file

```

**Author(s)**

J. H. Maindonald

**Examples**

```

## Not run:
xplot(data=coralRG$R, layout=coralRG$printer, FUN=imgplot)

## End(Not run)

## The function is currently defined as
function(data = DAAGbio::coralRG$R, images=1:6, layout = DAAGbio::coralRG$printer, mfrow=c(3,2),
         FUN = imgplot, device=NULL, title=NULL, width=7.5, height=10,
         paneltitles=c("1:R/G", "2:G/R", "3:R/G", "4:G/R", "5:R/G", "6:G/R"),
         paneltitles.line=0.5,
         mar=c(3.6,3.6,1.6,0.6), oma=c(0.6,0.6,1.6,0.6), file=NULL){
  if(is.null(title)){title <- as.character(substitute(data))
  title <- paste(title[2], title[3], sep=":")
  }
  if(is.null(file))file <- title
  nch <- nchar(title)
  if(!is.null(device)){devnam <- deparse(substitute(device))
  ext <- switch(devnam, ps="ps", pdf="pdf", png="png",
               jpeg="jpg", bitmap="bmp")
  file <- paste(title, ".", ext, sep="")
  print(file)
  device(file=file, width=width, height=height)
  }
  oldpar <- par(mfrow=mfrow, mgp=c(1,0.25,0), oma=oma, mar=mar)
  on.exit(par(oldpar))
  for(i in images){
    FUN(data[,i], layout=layout)
    mtext(side=3,line=paneltitles.line,paneltitles[i],adj=0)
  }
  mtext(side=3, line=0.25, title, outer=TRUE)
  if(!is.null(device))dev.off()
}

```

# Index

## \* datasets

- coralRG, [2](#)
- coralTargets, [2](#)
- DEnma, [3](#)
- plantStressCounts, [8](#)
- primateDNA, [10](#)

## \* hplot

- imgplot, [3](#)
- plotprintseq, [9](#)
- xplot, [11](#)

coralRG, [2](#)  
coralTargets, [2](#)

DEnma, [3](#)

imgplot, [3](#)

plantStressCounts, [8](#)  
plotprintseq, [9](#)  
primateDNA, [10](#)

xplot, [11](#)