

# Package ‘rafalib’

May 9, 2026

**Type** Package

**Version** 1.0.4

**Title** Convenience Functions for Routine Data Exploration

**Depends** R (>= 3.1.2),

**Imports** RColorBrewer, BiocManager

**Description** A series of shortcuts for routine tasks originally developed by Rafael A. Irizarry to facilitate data exploration.

**License** Artistic-2.0

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**NeedsCompilation** no

**Author** Rafael A. Irizarry [aut, cre],  
Michael I. Love [aut]

**Maintainer** Rafael A. Irizarry <rafael\_irizarry@dfci.harvard.edu>

**Repository** CRAN

**Date/Publication** 2025-04-08 04:00:02 UTC

## Contents

as.fumeric . . . . .	2
bartab . . . . .	2
imagemat . . . . .	3
imagesort . . . . .	4
install_bioc . . . . .	5
largeobj . . . . .	5
maplot . . . . .	6
mypar . . . . .	7
myplclust . . . . .	8
nullplot . . . . .	9
peek . . . . .	10
popstd . . . . .	10

popvar . . . . .	11
sboxplot . . . . .	11
shist . . . . .	12
splitit . . . . .	13
splot . . . . .	13
stripplot . . . . .	14

<b>Index</b>	<b>15</b>
--------------	-----------

---

as.fumeric	<i>converts to factor and then numeric</i>
------------	--

---

### Description

Converts a vector of characters into factors and then converts these into numeric.

### Usage

```
as.fumeric(x, levels = unique(x))
```

### Arguments

x	a character vector
levels	the levels to be used in the call to factor

### Author(s)

Rafael A. Irizarry

### Examples

```
group = c("a","a","b","b")
plot(seq_along(group),col=as.fumeric(group))
```

---

bartab	<i>bartab</i>
--------	---------------

---

### Description

Plot the overlap of three groups with a barplot

### Usage

```
bartab(x, y, z, names, skipNone = FALSE, ...)
```

**Arguments**

x                    logical  
 y                    logical  
 z                    logical  
 names                a character vector of length 3  
 skipNone            remove the "none" group  
 ...                   further arguments passed on to [barplot](#)

**Author(s)**

Michael I. Love

**Examples**

```
set.seed(1)
x <- sample(c(FALSE,TRUE), 10, replace=TRUE)
y <- sample(c(FALSE,TRUE), 10, replace=TRUE)
z <- sample(c(FALSE,TRUE), 10, replace=TRUE)
bartab(x,y,z,c("X","Y","Z"))
```

imagemat

*image of a matrix*

**Description**

Produces an image of a matrix which matches the natural orientation.

**Usage**

```
imagemat(
  x,
  col = colorRampPalette(c("white", "grey50"))(9),
  las = 1,
  xlab = "",
  ylab = "",
  ...
)
```

**Arguments**

x                    the matrix  
 col                  the colors  
 las                  as in par  
 xlab                x-axis title  
 ylab                y-axis title  
 ...                   arguments passed to image

**Author(s)**

Michael I. Love

**Examples**

```
x <- matrix(c(1,0,0,0,1,
              1,1,0,1,1,
              1,0,1,0,1,
              1,0,0,0,1,
              1,0,0,0,1),
            ncol=5,byrow=TRUE)

imagemat(x)
```

---

 imagesort

*image with sorted rows*


---

**Description**

the rows are sorted such that the first column has 2 blocks, the second column has 4 blocks, etc. see `example("imagesort")`

**Usage**

```
imagesort(x, col = c("white", "black"), ...)
```

**Arguments**

<code>x</code>	a matrix of 0s and 1s
<code>col</code>	the colors of 0 and 1
<code>...</code>	arguments to heatmap

**Author(s)**

Michael I. Love

**Examples**

```
x <- replicate(4,sample(0:1,40,TRUE))
imagesort(x)
```

---

`install_bioc`*Install or update Bioconductor and CRAN packages*

---

**Description**

This function is simply a wrapper for `link[BiocManager]{install}`. If `BiocManager` is not installed it is automatically installed.

**Usage**

```
install_bioc(...)
```

**Arguments**

... arguments passed on to `link[BiocManager]{install}`

**Details**

If `BiocManager` is installed you can simply call `BiocManager::install` instead.

**Author(s)**

Rafael A. Irizarry

**Examples**

```
install_bioc("affy")
```

---

`largeobj`*What are the largest objects in memory?*

---

**Description**

This function lists all the objects in the global environment and lists the `n` largest.

**Usage**

```
largeobj(n = 5, units = "Mb")
```

**Arguments**

`n` the number of objects to return  
`units` units to display, see `?object.size`

**Value**

a named character string of the size of the 'n' largest objects

**Author(s)**

Michael I. Love

**Examples**

```
x<-rnorm(10^5)
y<-rnorm(10^6)
z<-rnorm(2*10^6)
w<-rnorm(3*10^6)
largeobj(n=3)
```

---

maplot

*Bland Altman plot aka MA plot*

---

**Description**

Takes two vectors x and y and plots  $M=y-x$  versus  $A=(x+y)/2$ . If the vectors are more longer than length n the data is sampled to size n. A smooth curve is added to show trends.

**Usage**

```
maplot(
  x,
  y,
  n = 10000,
  subset = NULL,
  xlab = NULL,
  ylab = NULL,
  curve.add = TRUE,
  curve.col = 2,
  curve.span = 1/2,
  curve.lwd = 2,
  curve.n = 2000,
  ...
)
```

**Arguments**

x	a numeric vector
y	a numeric vector
n	a numeric value. If <code>length(x)</code> is larger than n, the x and y are sampled down.
subset	index of the points to be plotted

xlab	a title for the x axis
ylab	a title for the y axis
curve.add	if TRUE a smooth curve is fit to the data and displayed. The function <code>loess</code> is used to fit the curve.
curve.col	a numeric value that determines the color of the smooth curve
curve.span	is passed on to <code>loess</code> as the span argument
curve.lwd	the line width for the smooth curve
curve.n	a numeric value that determines the sample size used to fit the curve. This makes fitting the curve faster with large datasets
...	further arguments passed to <code>plot</code>

**Author(s)**

Rafael A. Irizarry

**Examples**

```
n <- 10000
signal <- runif(n,4,15)
bias <- (signal/5 - 2)^2
x <- signal + rnorm(n)
y <- signal + bias + rnorm(n)
maplot(x,y)
```

---

mypar

*mypar*

---

**Description**

Called without arguments, this function optimizes graphical parameters for the RStudio plot window. `bigpar` uses big fonts which are good for presentations.

**Usage**

```
mypar(
  a = 1,
  b = 1,
  brewer.n = 8,
  brewer.name = "Dark2",
  cex.lab = 1,
  cex.main = 1.2,
  cex.axis = 1,
  mar = c(2.5, 2.5, 1.6, 1.1),
  mgp = c(1.5, 0.5, 0),
  ...
)
```

**Arguments**

a	the first entry of the vector passed to mar
b	the second entry of the vector passed to mar
brewer.n	parameter n passed to <a href="#">brewer.pal</a>
brewer.name	parameters name passed to <a href="#">brewer.pal</a>
cex.lab	passed on to <a href="#">par</a>
cex.main	passed on to <a href="#">par</a>
cex.axis	passed on to <a href="#">par</a>
mar	passed on to <a href="#">par</a>
mgp	passed on to <a href="#">par</a>
...	other parameters passed on to <a href="#">par</a>

**Author(s)**

Rafael A. Irizarry

**Examples**

```
mypar()  
plot(cars)  
bigpar()  
plot(cars)
```

---

myplclust

*plclust in colour*

---

**Description**

Modification of plclust for plotting hclust objects in *\*in colour\**!

**Usage**

```
myplclust(  
  hclust,  
  labels = hclust$labels,  
  lab.col = rep(1, length(hclust$labels)),  
  hang = 0.1,  
  xlab = "",  
  sub = "",  
  ...  
)
```

**Arguments**

hclust	hclust object
labels	a character vector of labels of the leaves of the tree
lab.col	colour for the labels; NA=default device foreground colour
hang	as in <code>hclust</code> & <code>plclust</code>
xlab	title for x-axis (defaults to no title)
sub	subtitle (defaults to no subtitle)
...	further arguments passed to <code>plot</code>

**Author(s)**

Eva KF Chan

**Examples**

```
data(iris)
hc <- hclust( dist(iris[,1:4]) )
myplclust(hc, labels=iris$Species,lab.col=as.numeric(iris$Species))
```

---

nullplot

*nullplot*

---

**Description**

Make an plot with nothing in it

**Usage**

```
nullplot(x1 = 0, x2 = 1, y1 = 0, y2 = 1, xlab = "", ylab = "", ...)
```

**Arguments**

x1	lowest x-axis value
x2	largest x-axis value
y1	lowest y-axis value
y2	largest y-axis value
xlab	x-axis title, defaults to no title
ylab	y-axis title, defaults to no title
...	further arguments passed on to <code>plot</code>

**Examples**

```
nullplot()
```

peek *peek at the top of a text file*

---

### Description

this returns a character vector which shows the top n lines of a file. Note: I realized after the fact that this is essentially a duplicate of the base R function readLines.

### Usage

```
peek(x, n = 2)
```

### Arguments

x	a filename
n	the number of lines to return

### Author(s)

Michael I. Love

### Examples

```
filename <- tempfile()
x<-matrix(round(rnorm(10^4),2),1000,10)
colnames(x)=letters[1:10]
write.csv(x,file=filename,row.names=FALSE)
peek(filename)
```

---

popsd *population standard deviation*

---

### Description

Returns the population standard deviation. Note that `sd` returns the unbiased sample estimate of the population standard deviation. It simply multiplies the result of `var` by  $(n-1) / n$  with  $n$  the population size and takes the square root.

### Usage

```
popsd(x, na.rm = FALSE)
```

### Arguments

x	a numeric vector or an R object which is coercible to one by <code>as.vector(x, "numeric")</code> .
na.rm	logical. Should missing values be removed?

---

popvar	<i>population variance</i>
--------	----------------------------

---

**Description**

Returns the population variance. Note that `var` returns the unbiased sample estimate of the population variance. It simply multiplies the result of `var` by  $(n-1) / n$  with  $n$  the population size.

**Usage**

```
popvar(x, ...)
```

**Arguments**

<code>x</code>	a numeric vector, matrix or data frame.
<code>...</code>	further arguments passed along to <code>var</code>

**Examples**

```
x <- c(0,1) ##variance should be 0.5^2=0.25
var(x)
popvar(x)
```

---

sboxplot	<i>smart boxplot</i>
----------	----------------------

---

**Description**

draws points or boxes depending on sample size

**Usage**

```
sboxplot(x, ...)
```

**Arguments**

<code>x</code>	a named list of numeric vectors
<code>...</code>	further arguments passed on to <code>boxplot</code>

**Examples**

```
sboxplot(list(a=rnorm(15),b=rnorm(75),c=rnorm(10000)))
```

shist

*smooth histogram***Description**

a smooth histogram with unit indicator (we're simply scaling the kernel density estimate). The advantage of this plot is its interpretability since the height of the curve represents the frequency of a interval of size `unit` around the point in question. Another advantage is that if `z` is a matrix, curves are plotted together.

**Usage**

```
shist(
  z,
  unit,
  bw = "nrd0",
  n,
  from,
  to,
  plotHist = FALSE,
  add = FALSE,
  xlab,
  ylab = "Frequency",
  xlim,
  ylim,
  main,
  ...
)
```

**Arguments**

<code>z</code>	the data
<code>unit</code>	the unit which determines the y axis scaling and is drawn
<code>bw</code>	arguments to density
<code>n</code>	arguments to density
<code>from</code>	arguments to density
<code>to</code>	arguments to density
<code>plotHist</code>	a logical: should an actual histogram be drawn under curve?
<code>add</code>	a logical: add should the curve be added to existing plot?
<code>xlab</code>	x-axis title, defaults to no title
<code>ylab</code>	y-axis title, defaults to no title
<code>xlim</code>	range of the x-axis
<code>ylim</code>	range of the y-axis
<code>main</code>	an overall title for the plot: see <a href="#">title</a> .
<code>...</code>	arguments to lines

**Examples**

```
set.seed(1)
x = rnorm(50)
par(mfrow=c(2,1))
hist(x, breaks=-5:5)
shist(x, unit=1, xlim=c(-5,5))
```

---

splitit

*split it*

---

**Description**

Creates an list of indexes for each unique entry of x

**Usage**

```
splitit(x)
```

**Arguments**

x                    a vector

**Examples**

```
x <- c("a", "a", "b", "a", "b", "c", "b", "b")
splitit(x)
```

---

split

*smart plot*

---

**Description**

if n > 10,000, make a random subset of 10,000 and plot. You can also specify a specific subset to plot. If length of subset is larger than n, a random sample is still used to reduce data size.

**Usage**

```
split(x, y, n = 10000, subset = NULL, xlab = NULL, ylab = NULL, ...)
```

**Arguments**

x	the x data
y	the y data
n	the number to subset
subset	explicit subset index (optional).
xlab	title for the x-axis
ylab	title for the y-axis
...	further parameters passed on to plot

**Examples**

```
x <- rnorm(1e5)
y <- rnorm(1e5)
stripplot(x,y,pch=16,col=rgb(0,0,0,.25))
```

---

stripplot

*Better defaults for stripchart*

---

**Description**

This simply calls `stripchart` but specifies a vertical plot with jitter and using `pch=1`.

**Usage**

```
stripplot(...)
```

**Arguments**

... passed to `stripchart`

**Value**

a plot

# Index

as.fumeric, 2

barplot, 3  
bartab, 2  
bigpar (mypar), 7  
boxplot, 11  
brewer.pal, 8

hclust, 9

imagemat, 3  
imagesort, 4  
install\_bioc, 5

largeobj, 5  
loess, 7

maplot, 6  
mypar, 7  
myplclust, 8

nullplot, 9

par, 8  
peek, 10  
plclust, 9  
plot, 7, 9  
popsd, 10  
popvar, 11

sboxplot, 11  
sd, 10  
shist, 12  
splitit, 13  
splot, 13  
striplot, 14

title, 12

var, 10, 11