Base Graphics in R (part I)

STAT 133

Gaston Sanchez

R Graphics

2 main graphics systems

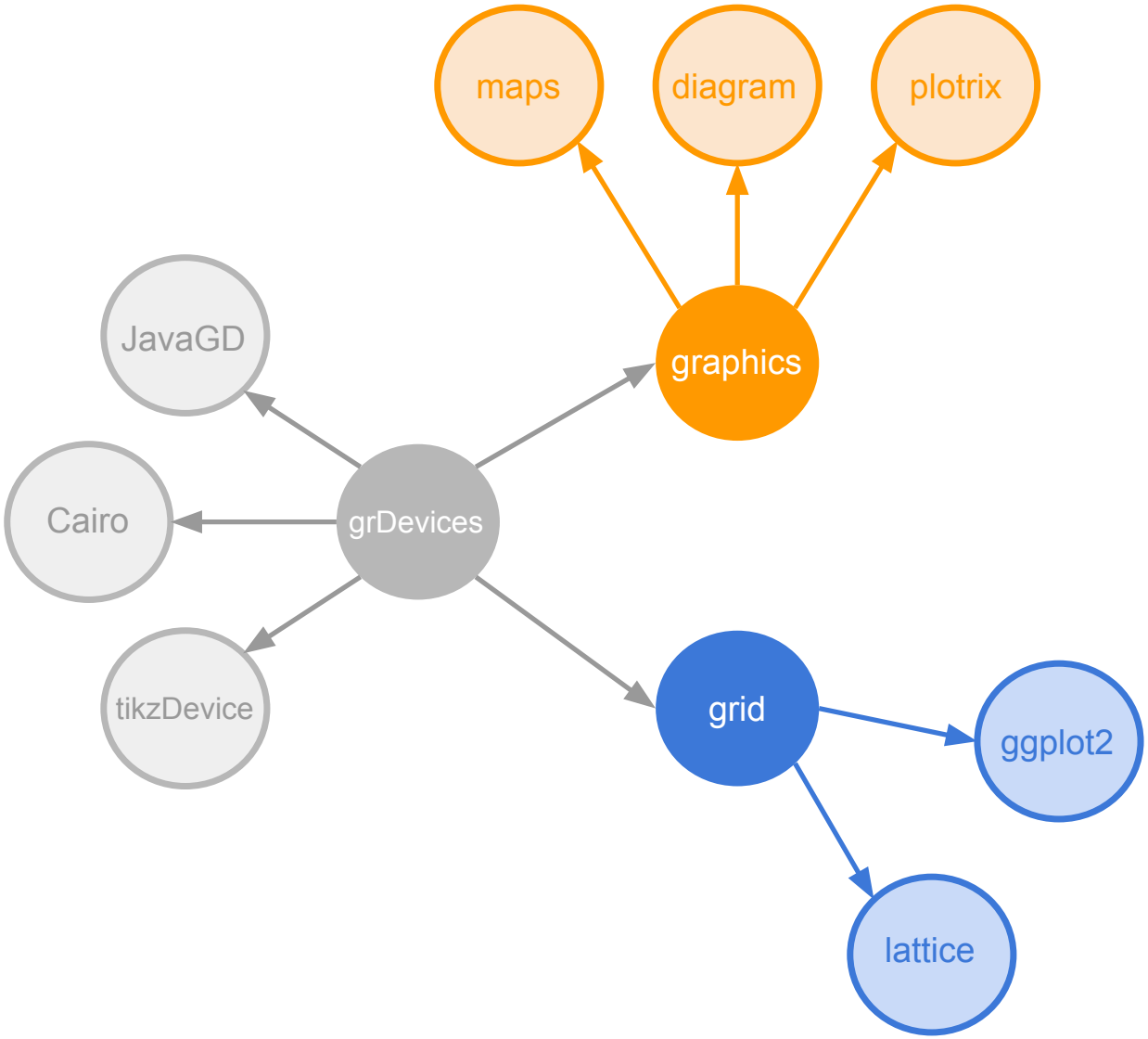
`"graphics"` & `"grid"`

Graphics Systems

- `"graphics"` and `"grid"` are the two main graphics systems in R
- `"graphics"` is the *traditional* system, also referred to as *base graphics*
- `"grid"` prodives low-level functions for programming plotting functions

Graphics Engine

- Underneath `"graphics"` and `"grid"` there is the package `"grDevices"`
- `"grDevices"` is the graphics **engine** in R
- It provides the graphics devices and support for colors and fonts


maps
diagram
plotrix
JavaGD
graphics
Cairo
grDevices
tikzDevice
grid
ggplot2
lattice

Package "graphics"

The package "graphics" is the traditional system; it provides functions for complete plots, as well as low-level facilities.

Many other graphics packages are built on top of graphics like "maps", "diagram", "pixmap", and many more.

Package "grid"

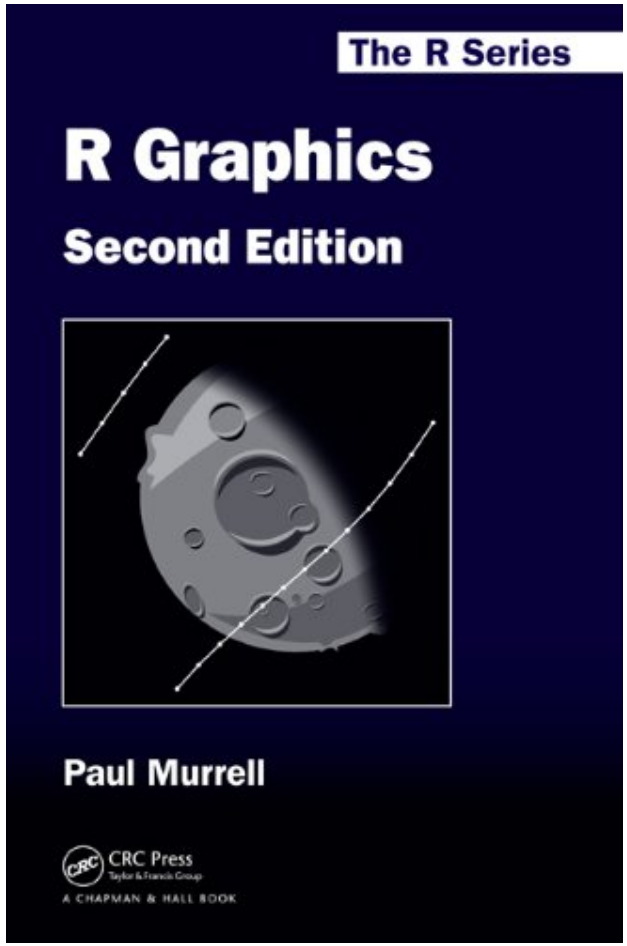
The "grid" package does not provide functions for drawing complete plots.

"grid" is not used directly to produce statistical plots. Instead, it is used to build other graphics packages like "lattice" or "ggplot2".

In this course

- We'll focus on the packages `"graphics"` and `"ggplot2"`
- `"graphics"` is the traditional plotting system in R, and many functions and packages are built on top of it.
- `"ggplot2"` excels at providing graphics for visualizing multivariate data sets —in `data.frame` format—, while taking care of many issues for superior visual displays.

R Graphics by Paul Murrell



Some Resources

- **R Graphics** by Paul Murrell
  book and webpage
- **R Graphics Cookbook** by Winston Chang
  http://www.cookbook-r.com/Graphs/
- **R Graphs Cookbook** by Hrishi Mittal
- **Graphics for Statistics and Data Analysis with R** by Kevin Keen

Traditional (Base) Graphics

Types of graphics functions

Graphics functions can be divided into two main types:

- **high-level** functions produce complete plots, e.g. `barplot()`, `boxplot()`, `dotchart()`
- **low-level** functions add further output to an existing plot, e.g. `text()`, `points()`, `legend()`

The `plot()` function

- `plot()` is the most important high-level function in traditional graphics
- The first argument to `plot()` provides the data to plot
- The provided data can take different forms: e.g. vectors, factors, matrices, data frames.
- To be more precise, `plot()` is a generic function
- You can create your own `plot()` method function

Basic Plots with `plot()`

In its basic form, we can use `plot()` to make graphics of:

- one single variable
- two variables
- multiple variables

Plots of One Variable

High-level graphics of a single variable

Function	Data	Description
`plot()`	numeric	scatterplot
`plot()`	factor	barplot
`plot()`	1-D table	barplot

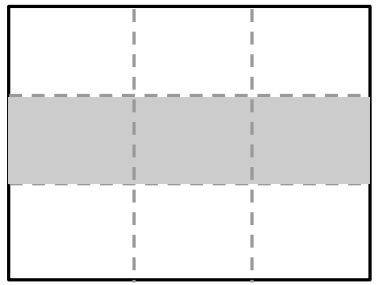
`numeric` can be either a vector or a 1-D array (e.g. row or column from a matrix)

One variable objects

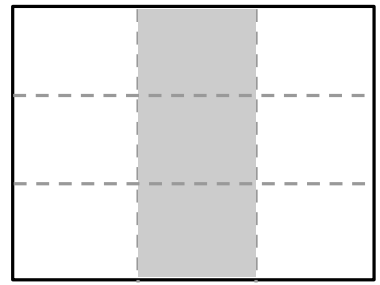
Vector / Factor

1-D table

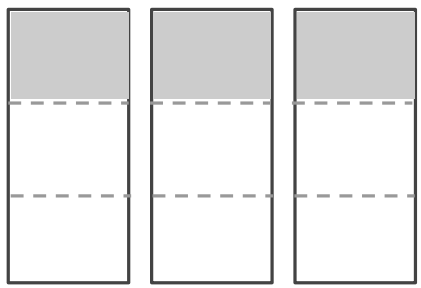
row (matrix)



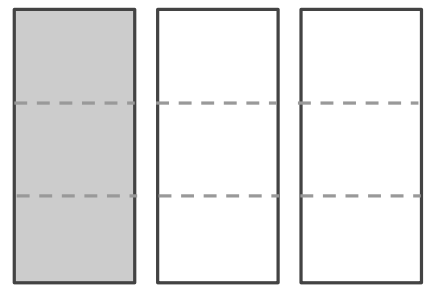
column (matrix)



row (data.frame)



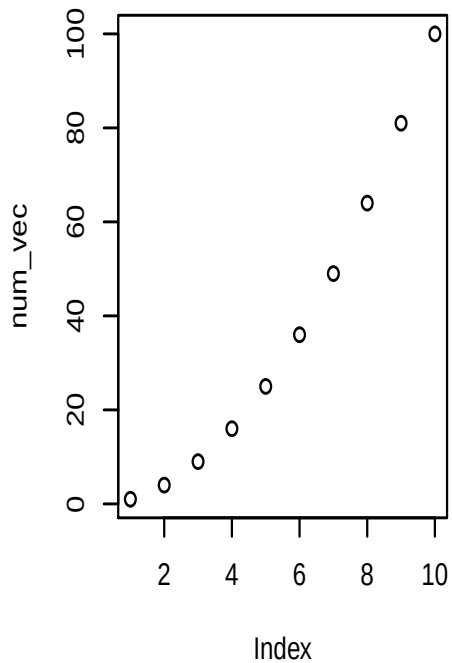
column (data.frame)

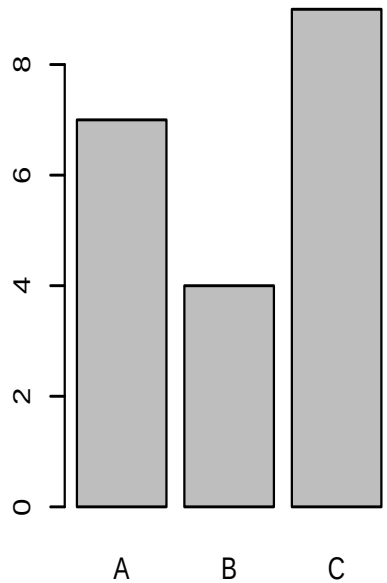


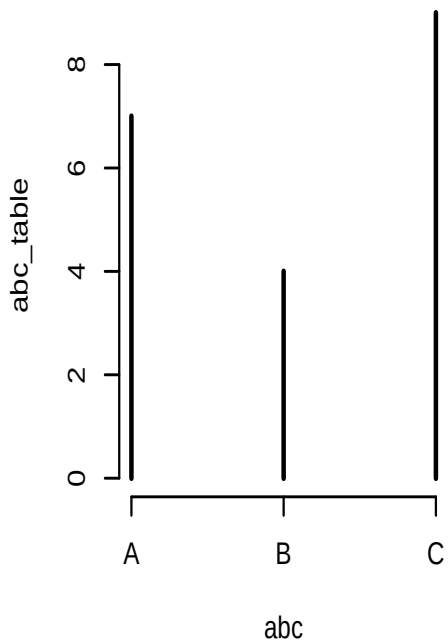
plot() of one variable

```r
# plot numeric vector
num_vec <- (c(1:10))^2
plot(num_vec)

# plot factor
set.seed(4)
abc <- factor(sample(c('A', 'B', 'C'), 20, replace = TRUE))
plot(abc)

# plot 1D-table
abc_table <- table(abc)
plot(abc_table)
```

`plot()` of one variable







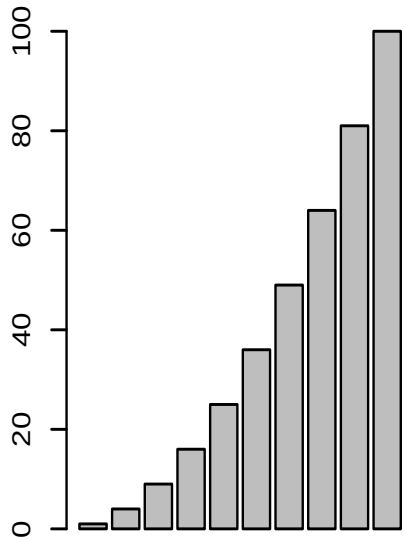
More high-level graphics of a single variable

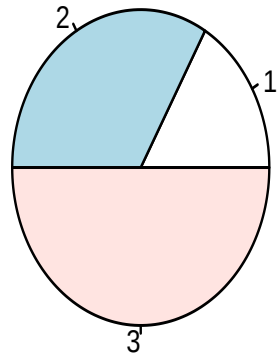
Function	Data	Description
`barplot()`	numeric	barplot
`pie()`	numeric	pie chart
`dotchart()`	numeric	dotplot
`boxplot()`	numeric	boxplot
`hist()`	numeric	histogram
`stripchart()`	numeric	1-D scatterplot
`stem()`	numeric	stem-and-leaf plot

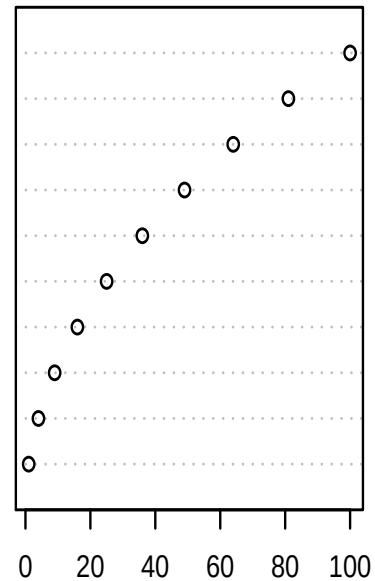
Plots of one variable

```r
# barplot numeric vector
barplot(num_vec)

# pie chart
pie(1:3)

# dot plot
dotchart(num_vec)
```

Plots of one variable





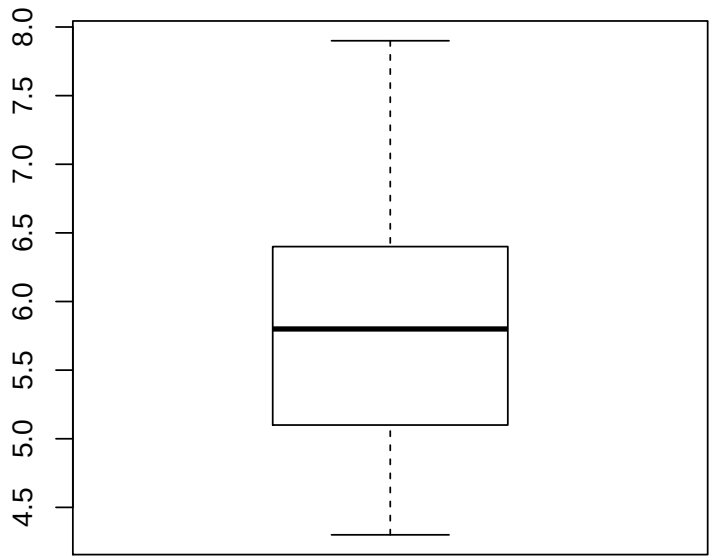


Plots of one variable

```
# barplot numeric vector
boxplot(num_vec)

# pie chart
hist(num_vec)

# dot plot
stripchart(num_vec)

# stem-and-leaf
stem(num_vec)
```

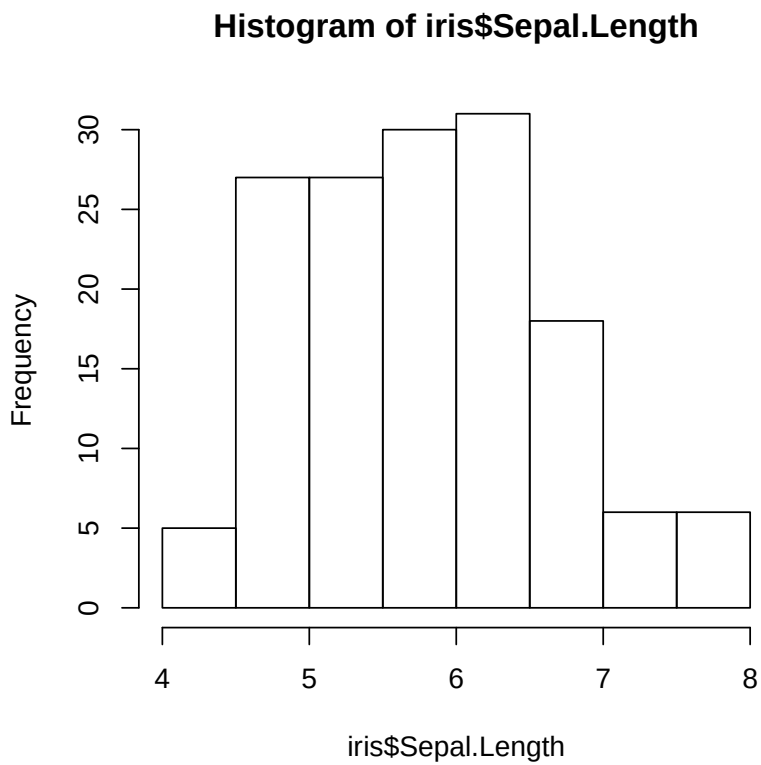
boxplot()

```
# boxplot
boxplot(iris$Sepal.Length)
```



hist()

```r
# histogram
hist(iris$Sepal.Length)
```



Test your knowledge

What option does not apply to histograms:

A) adjacent bars (no gaps)

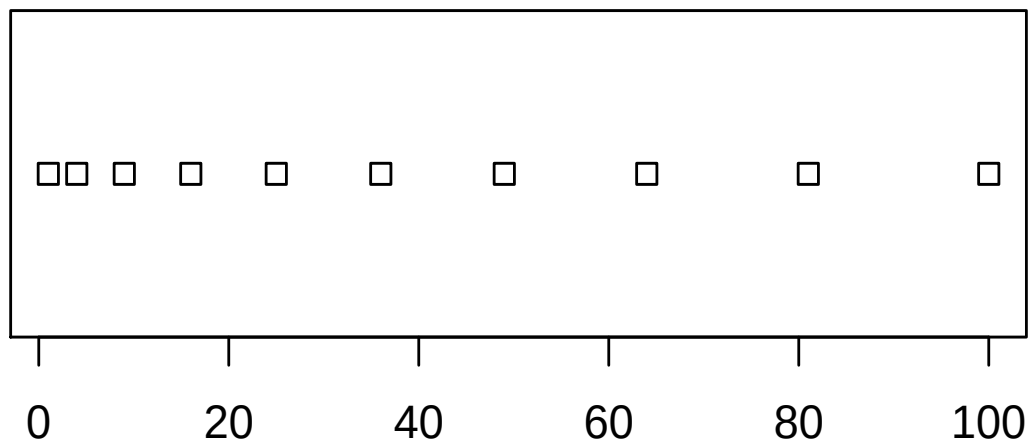
B) area of bars indicate proportions

C) bins of equal length

D) bars can be reordered

stripchart()

```
# strip-chart (1-D scatter plot)
# (for small sample sizes)
stripchart(num_vec)
```



stem()

```
# stem-and-leaf plot
# (for small sample sizes)
stem(num_vec)
```

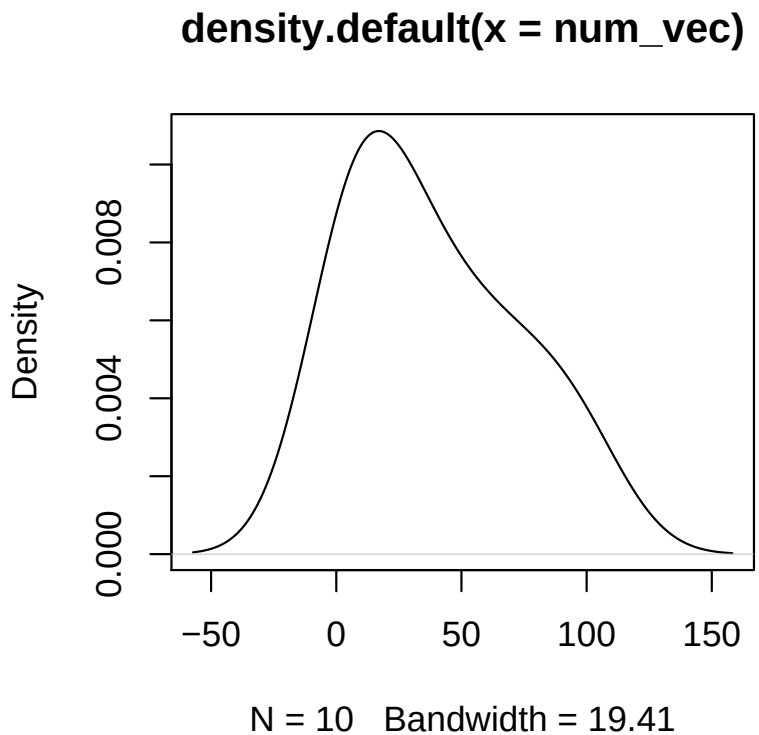
```
##
##   The decimal point is 1 digit(s) to the right of the |
##
##    0 | 1496
##    2 | 56
##    4 | 9
##    6 | 4
##    8 | 1
##   10 | 0
```

Kernel Density Curve

- Surprisingly, R does not have a specific function to plot density curves
- R does have the `density()` function which computes a kernel density estimate
- We can pass a `"density"` object to `plot()` in order to get a density curve.

Kernel Density Curve

```
# kernel density curve
dens <- density(num_vec)
plot(dens)
```



Test your knowledge

What type of plot is based on the five-number summary

A) bar chart

B) box plot

C) histogram

D) scatterplot

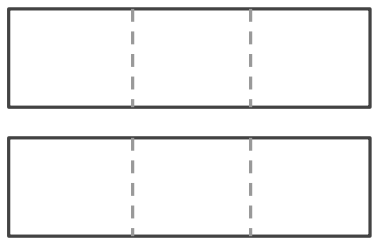
Plots of Two Variables

High-level graphics of two variables

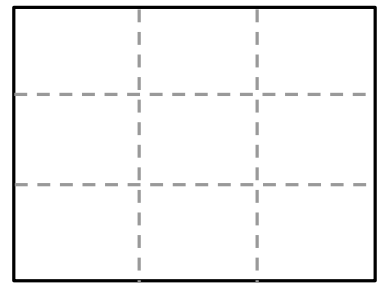
Function	Data	Description
`plot()`	numeric, numeric	scatterplot
`plot()`	numeric, factor	stripcharts
`plot()`	factor, numeric	boxplots
`plot()`	factor, factor	spineplot
`plot()`	2-column numeric matrix	scatterplot
`plot()`	2-column numeric data.frame	scatterplot
`plot()`	2-D table	mosaicplot

Two variable objects

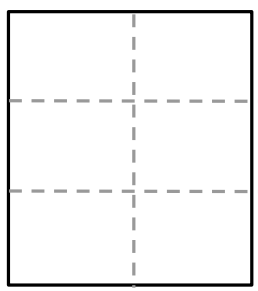
2 numeric vectors
num vector, factor
factor, num vector
2 factors



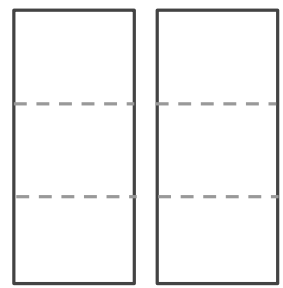
2-D table
(frequency or
crosstable)



2-column
(numeric matrix)



2-column
(numeric data.frame)

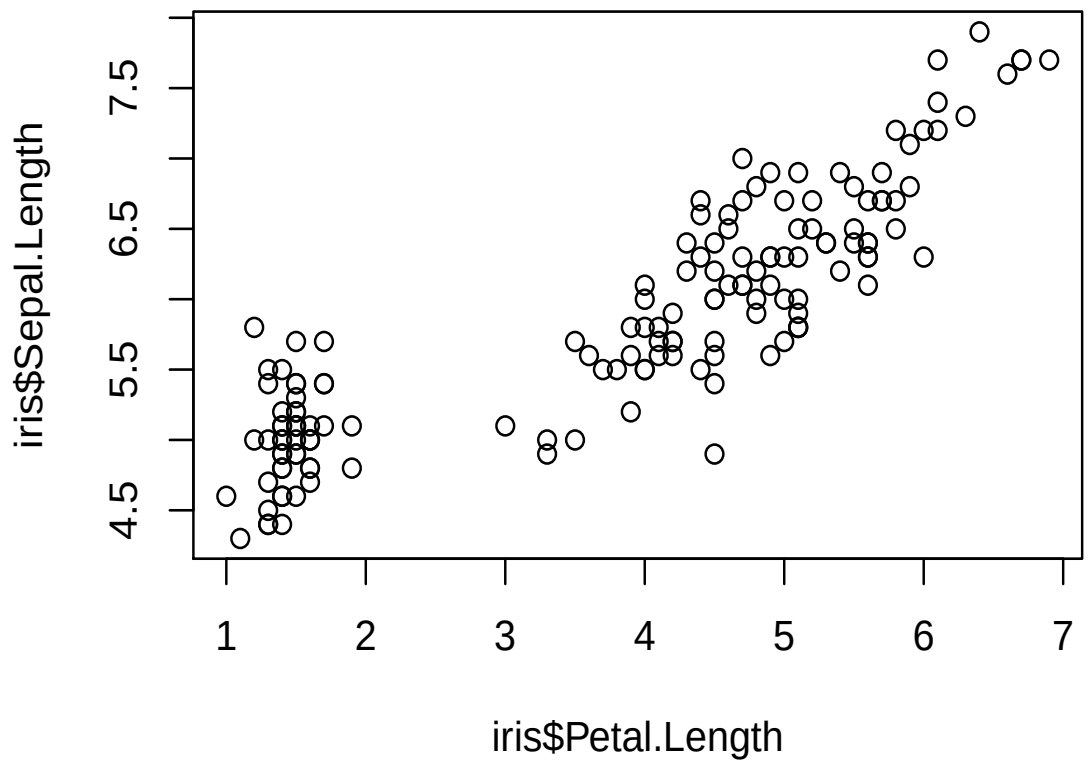


Plots of two variables

```r
# plot numeric, numeric
plot(iris$Petal.Length, iris$Sepal.Length)

# plot numeric, factor
plot(iris$Petal.Length, iris$Species)

# plot factor, numeric
plot(iris$Species, iris$Petal.Length)

# plot factor, factor
plot(iris$Species, iris$Species)
```

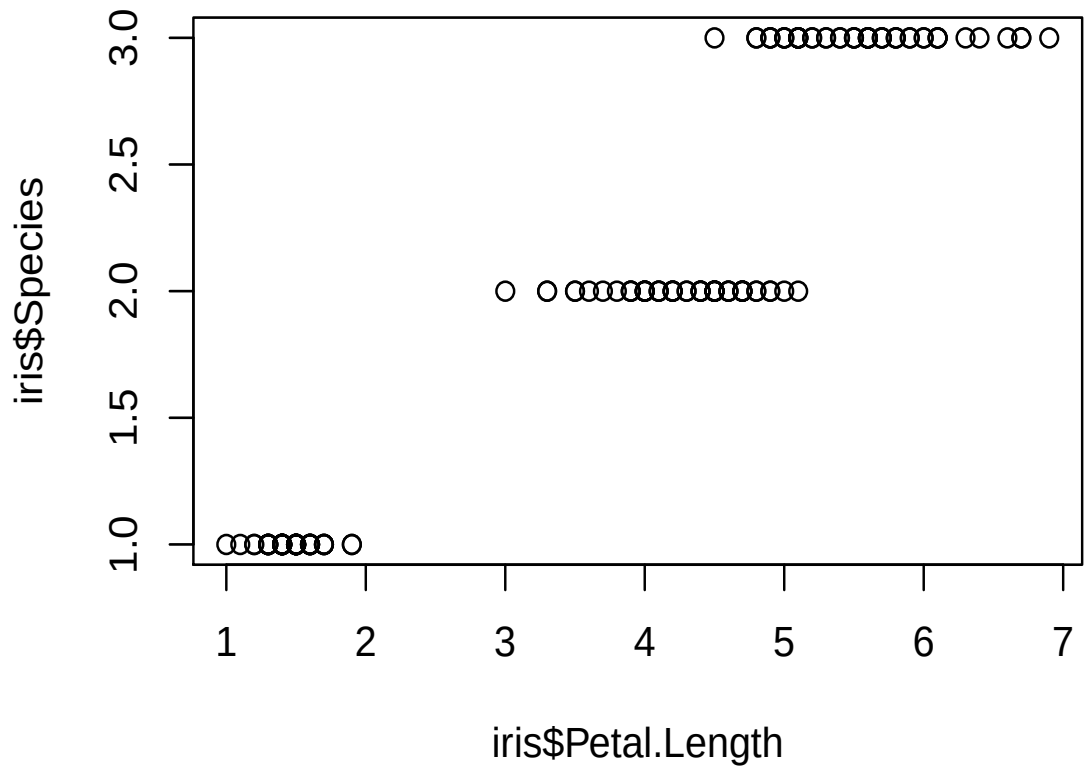
Plots of two variables

```
# plot numeric, numeric
plot(iris$Petal.Length, iris$Sepal.Length)
```



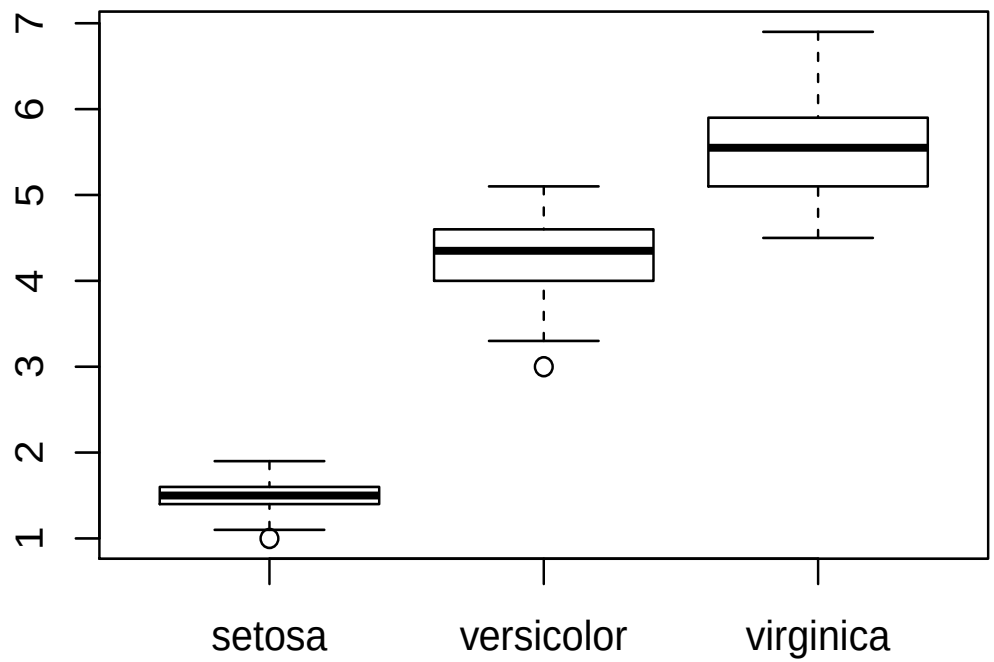
Plots of two variables

```
# plot numeric, factor
plot(iris$Petal.Length, iris$Species)
```



Plots of two variables

```
# plot factor, numeric
plot(iris$Species, iris$Petal.Length)
```



Plots of two variables

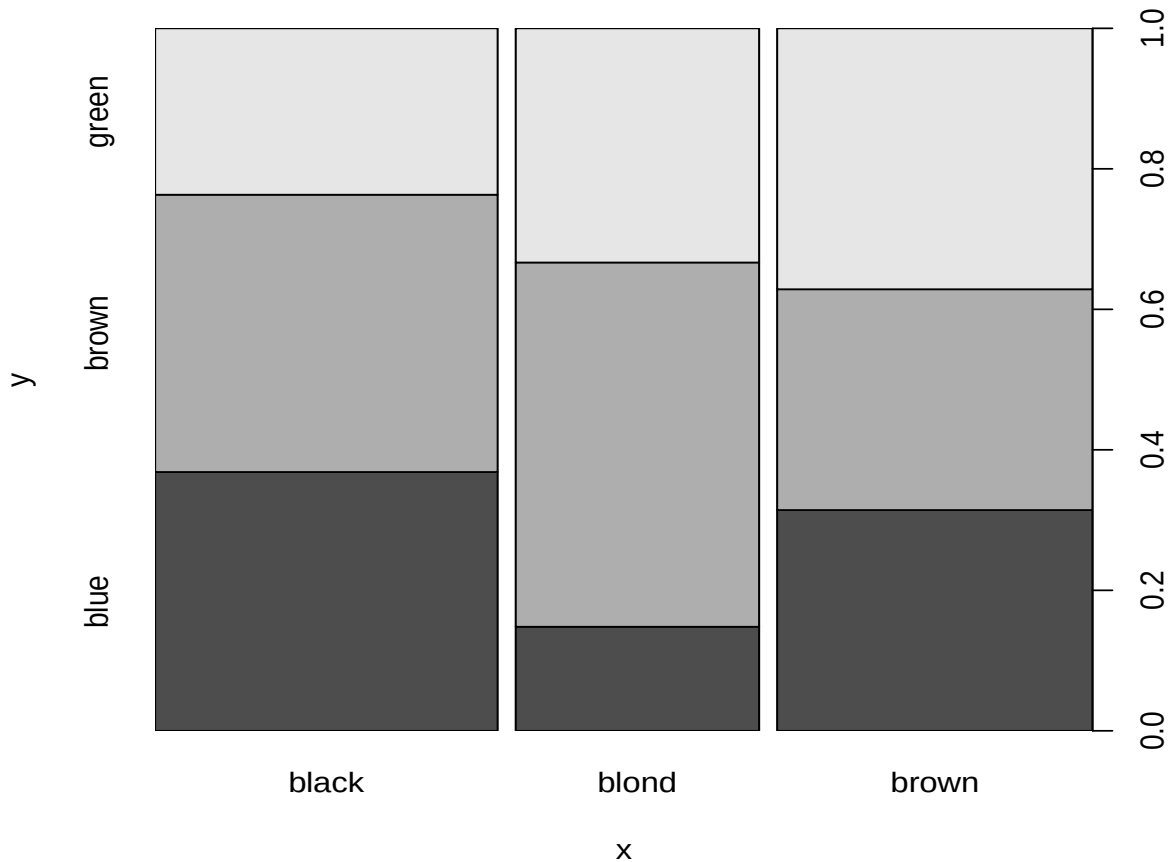
```
# plot factor, factor
plot(iris$Species, iris$Species)
```

Plots of two variables

```r
# some fake data
set.seed(1)

# hair color
hair <- factor(
  sample(c('blond', 'black', 'brown'), 100, replace = TRUE))

# eye color
eye <- factor(
  sample(c('blue', 'brown', 'green'), 100, replace = TRUE))
```

Plots of two variables

```
# plot factor, factor
plot(hair, eye)
```

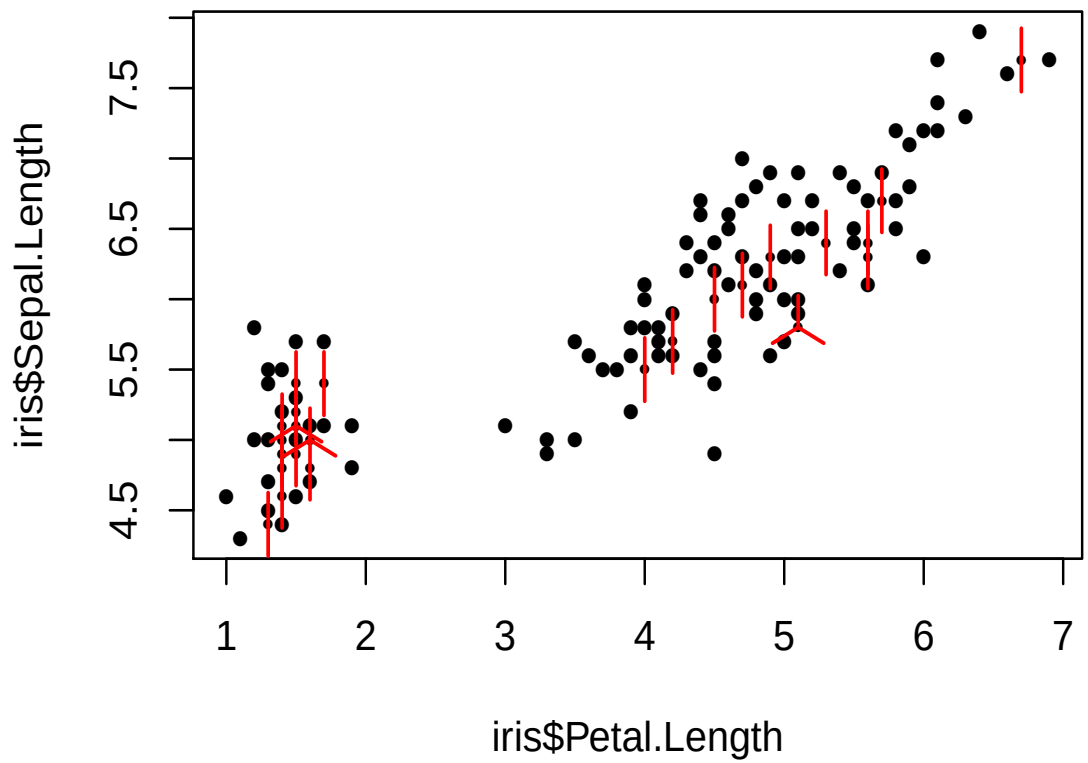


More high-level graphics of two variables

Function	Data	Description
`sunflowerplot()`	numeric, numeric	sunflower scatterplot
`smoothScatter()`	numeric, numeric	smooth scatterplot
`boxplot()`	list of numeric	boxplots
`barplot()`	matrix	stacked / side-by-side barplot
`dotchart()`	matrix	dotplot
`stripchart()`	list of numeric	stripcharts
`spineplot()`	numeric, factor	spinogram
`cdplot()`	numeric, factor	conditional density plot
`fourfoldplot()`	2x2 table	fourfold display
`assocplot()`	2-D table	association plot
`mosaicplot()`	2-D table	mosaic plot

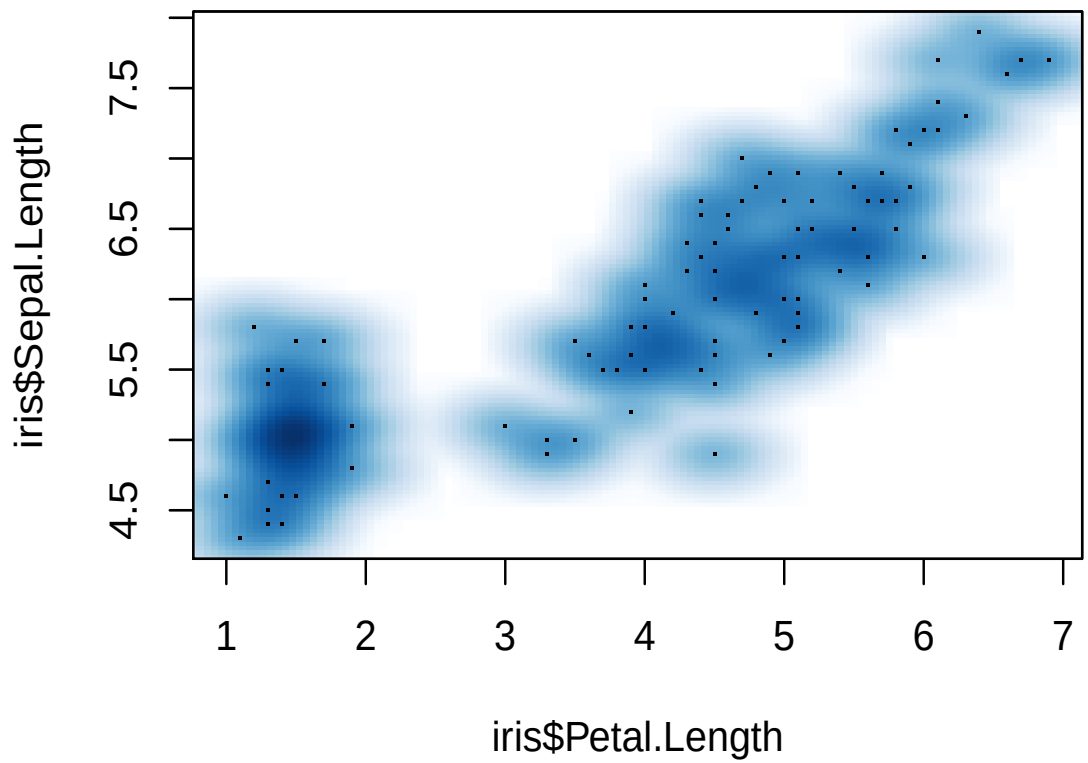
Plots of two variables

```
# sunflower plot (numeric, numeric)
sunflowerplot(iris$Petal.Length, iris$Sepal.Length)
```



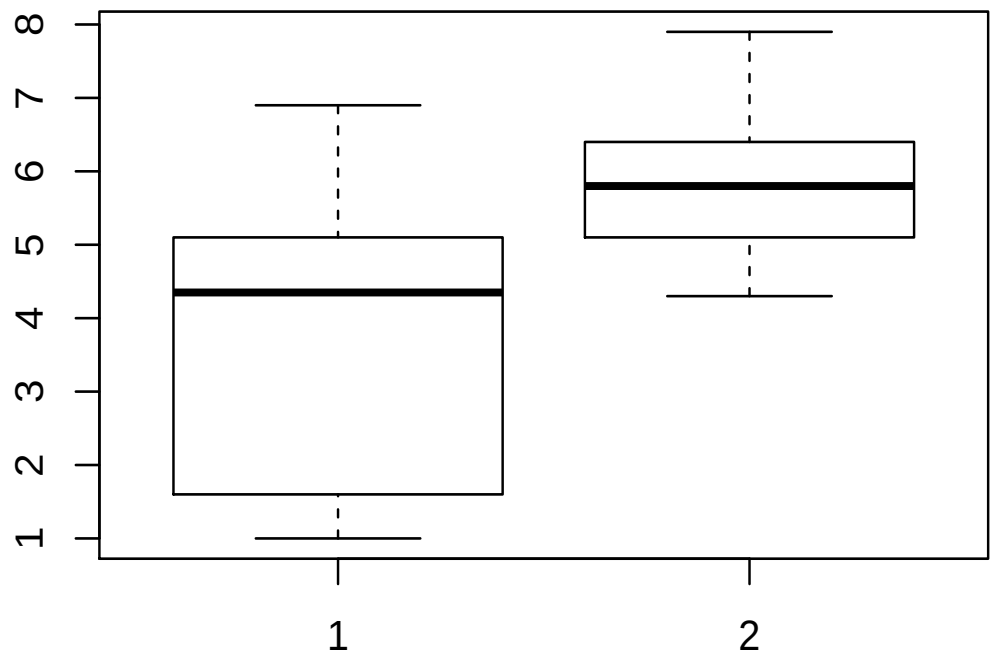
Plots of two variables

```
# smooth scatter plot (numeric, numeric)
smoothScatter(iris$Petal.Length, iris$Sepal.Length)
```



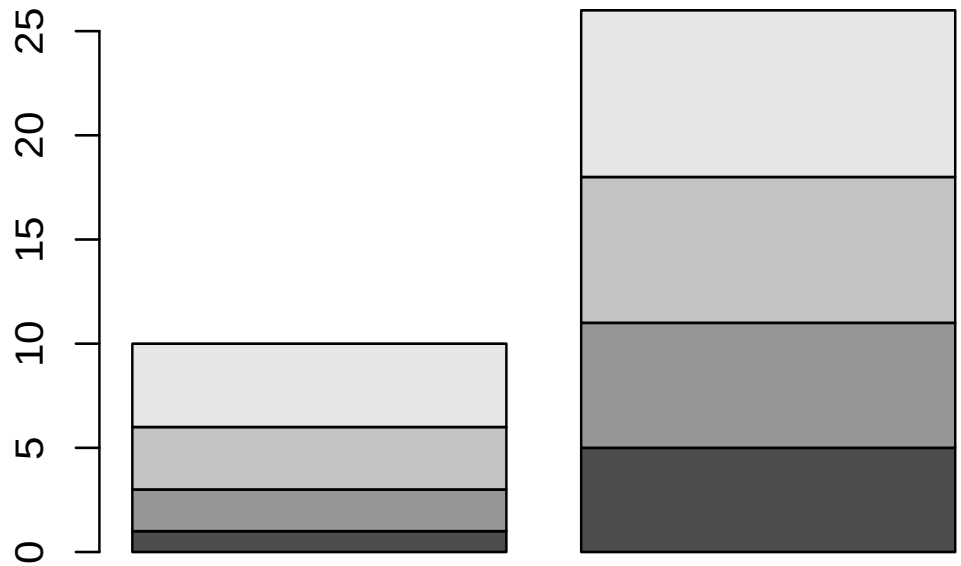
Plots of two variables

```
# boxplots (numeric, numeric)
boxplot(iris$Petal.Length, iris$Sepal.Length)
```



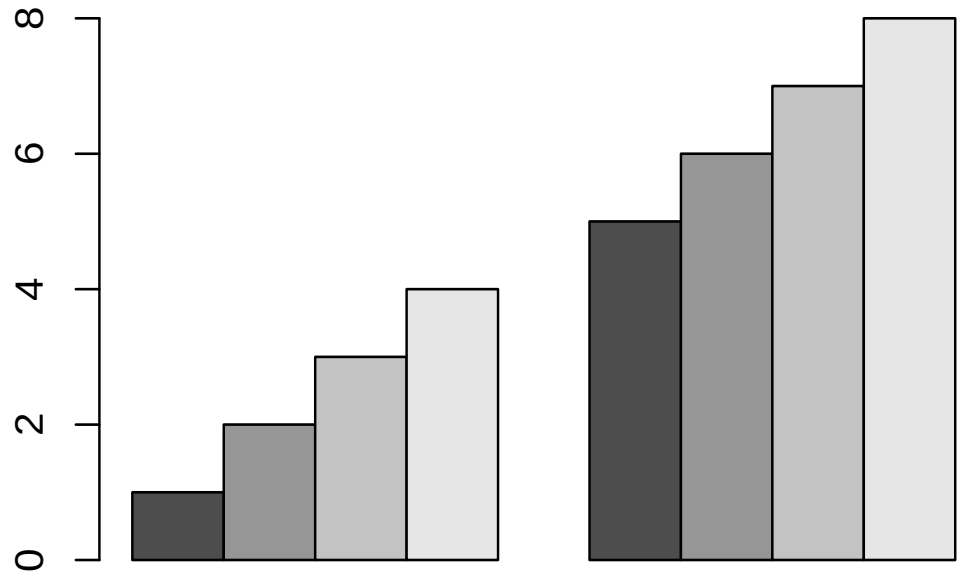
Plots of two variables

```r
m <- matrix(1:8, 4, 2)
# barplot (numeric matrix)
barplot(m)
```



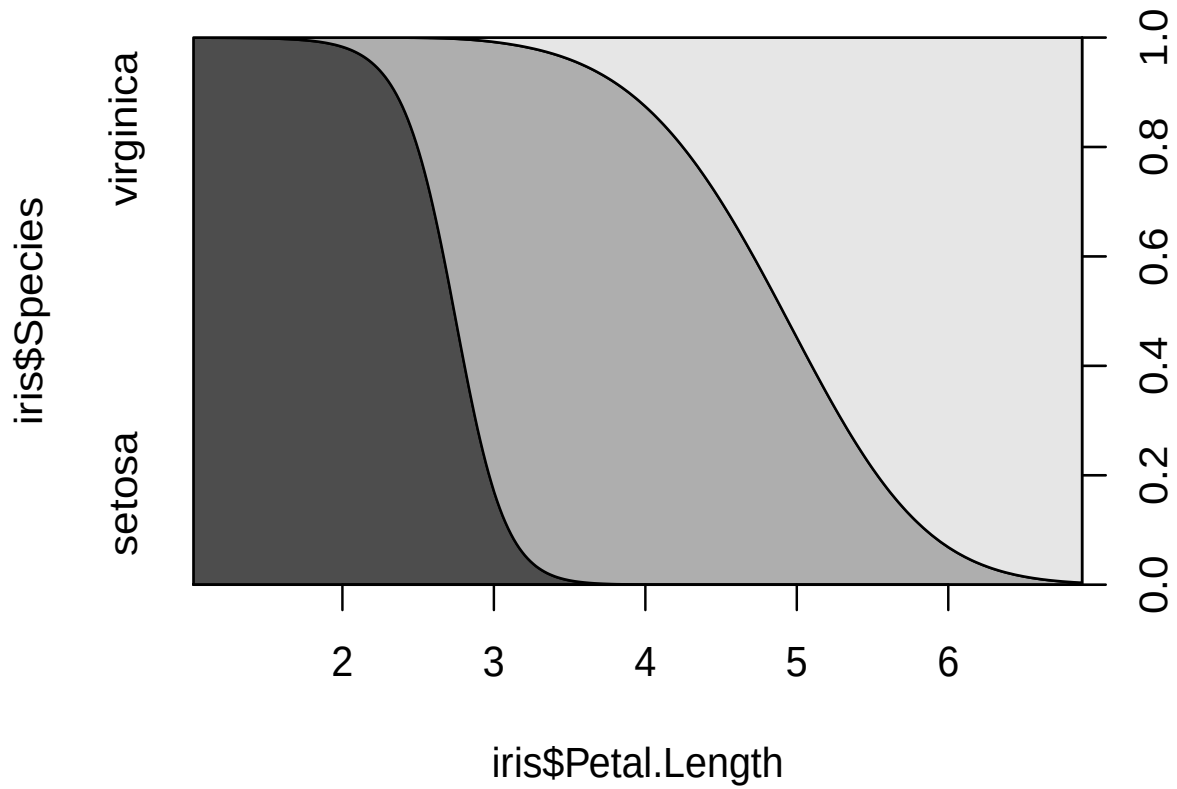
Plots of two variables

```r
m <- matrix(1:8, 4, 2)
# barplot (numeric matrix)
barplot(m, beside = TRUE)
```



Plots of two variables

```
# conditional density plot (numeric, factor)
cdplot(iris$Petal.Length, iris$Species)
```



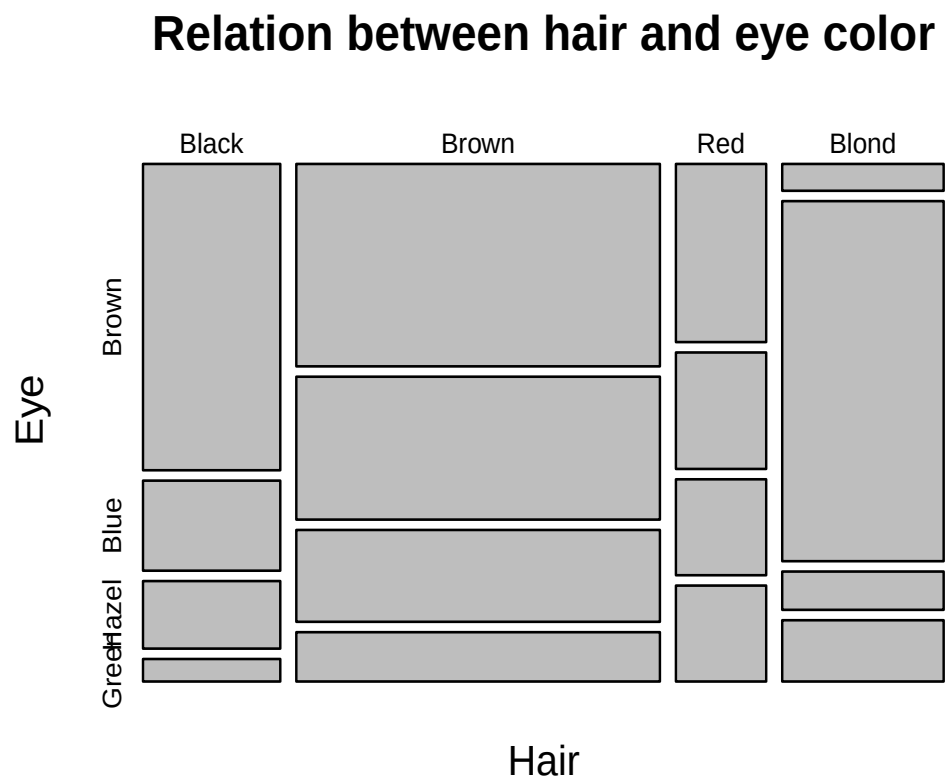
Two categorical variables: frequency table

```
# 2-D table (HairEyeColor data)
x <- margin.table(HairEyeColor, c(1, 2))
x
```

```
##         Eye
## Hair     Brown Blue Hazel Green
##   Black     68   20    15     5
##   Brown    119   84    54    29
##   Red       26   17    14    14
##   Blond      7   94    10    16
```

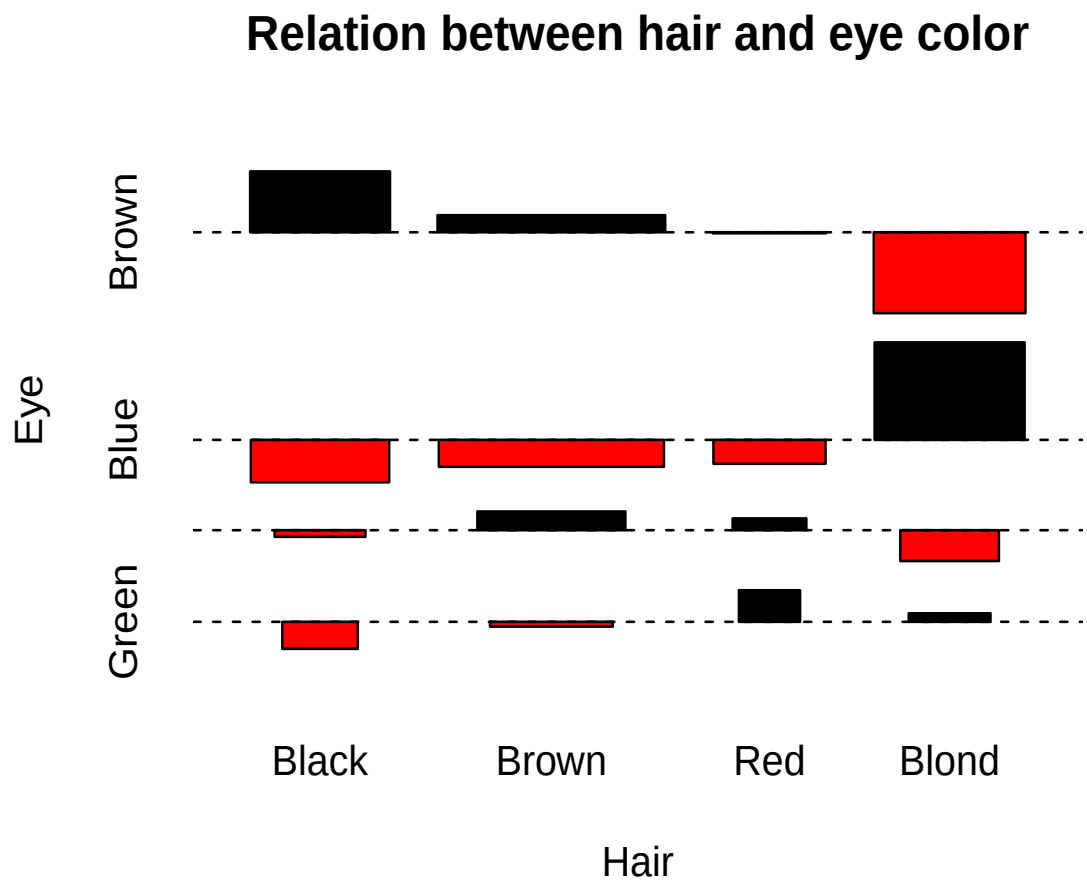
Plots of two categorical variables

```
# mosaic plot (2-D table)
mosaicplot(x, main = "Relation between hair and eye color")
```



Plots of two categorical variables

```
# association plot (2-D table)
assocplot(x, main = "Relation between hair and eye color")
```



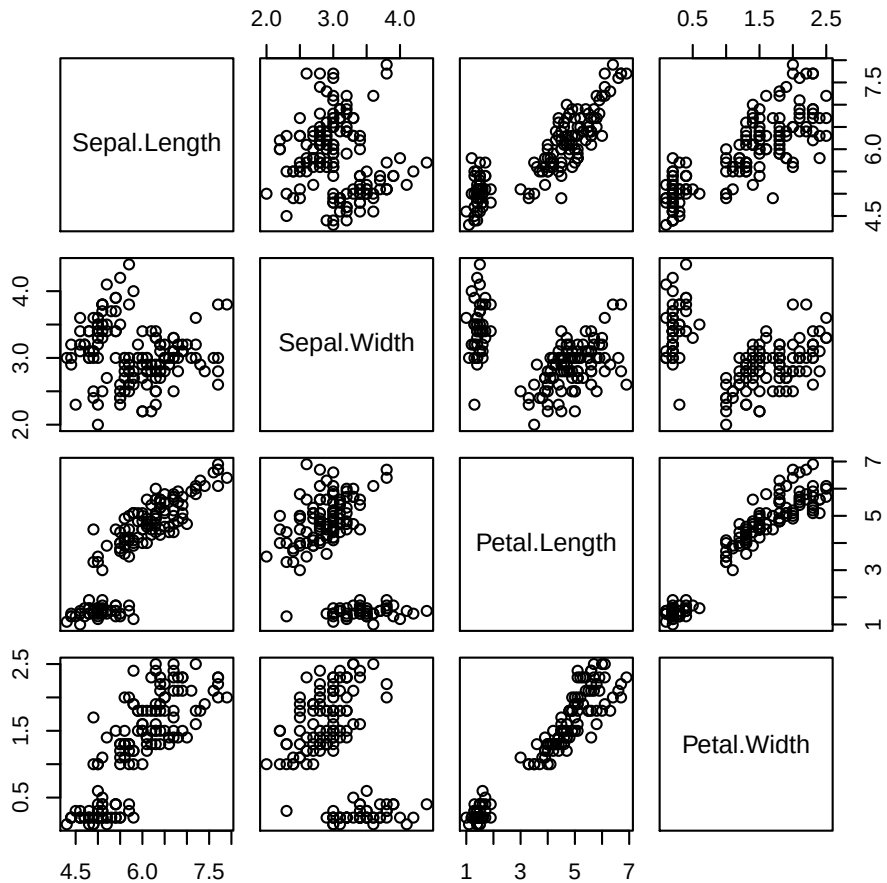
Plots of Multiple Variables

High-level graphics of multiple variables

Function	Data	Description
`plot()`	data frame	scatterplot matrix
`pairs()`	matrix	scatterplot matrix
`matplot()`	matrix	scatterplot
`stars()`	matrix	star plots
`image()`	numeric, numeric, numeric	image plot
`contour()`	numeric, numeric, numeric	contour plot
`filled.contour()`	numeric, numeric, numeric	filled contour plot
`persp()`	numeric, numeric, numeric	3-D surface
`symbols()`	numeric, numeric, numeric	symbols scatterplot
`mosaicplot()`	N-D table	mosaic plot

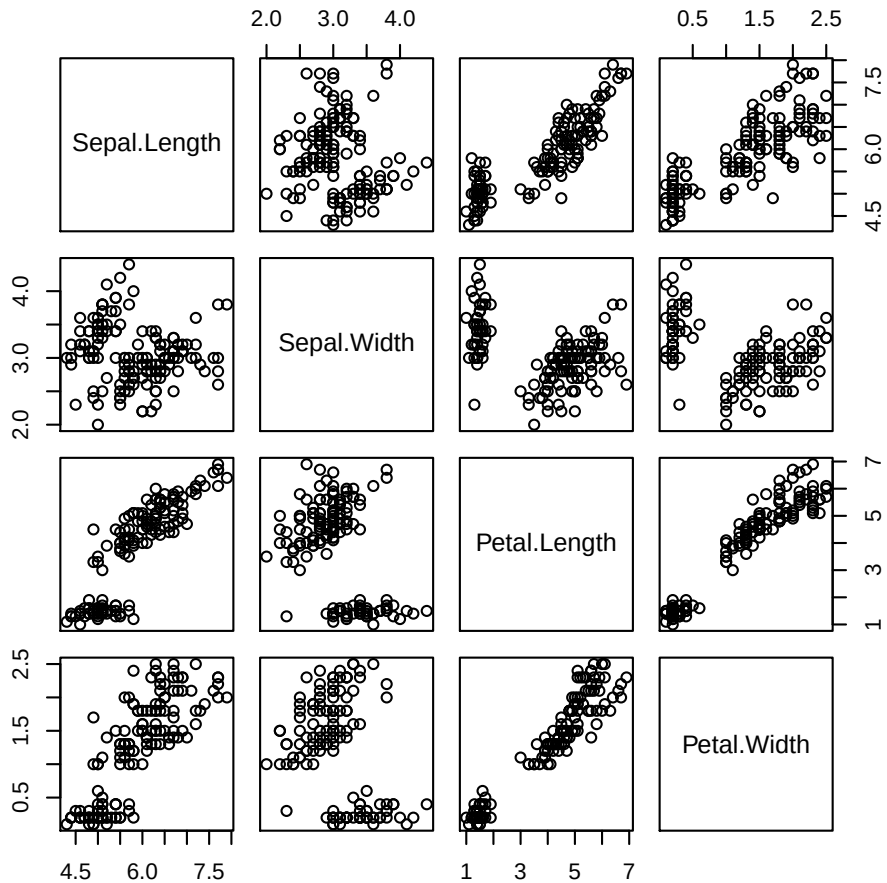
Plots of multiple variables

```
# scatter plot matrix (data frame)
plot(iris[ , 1:4])
```



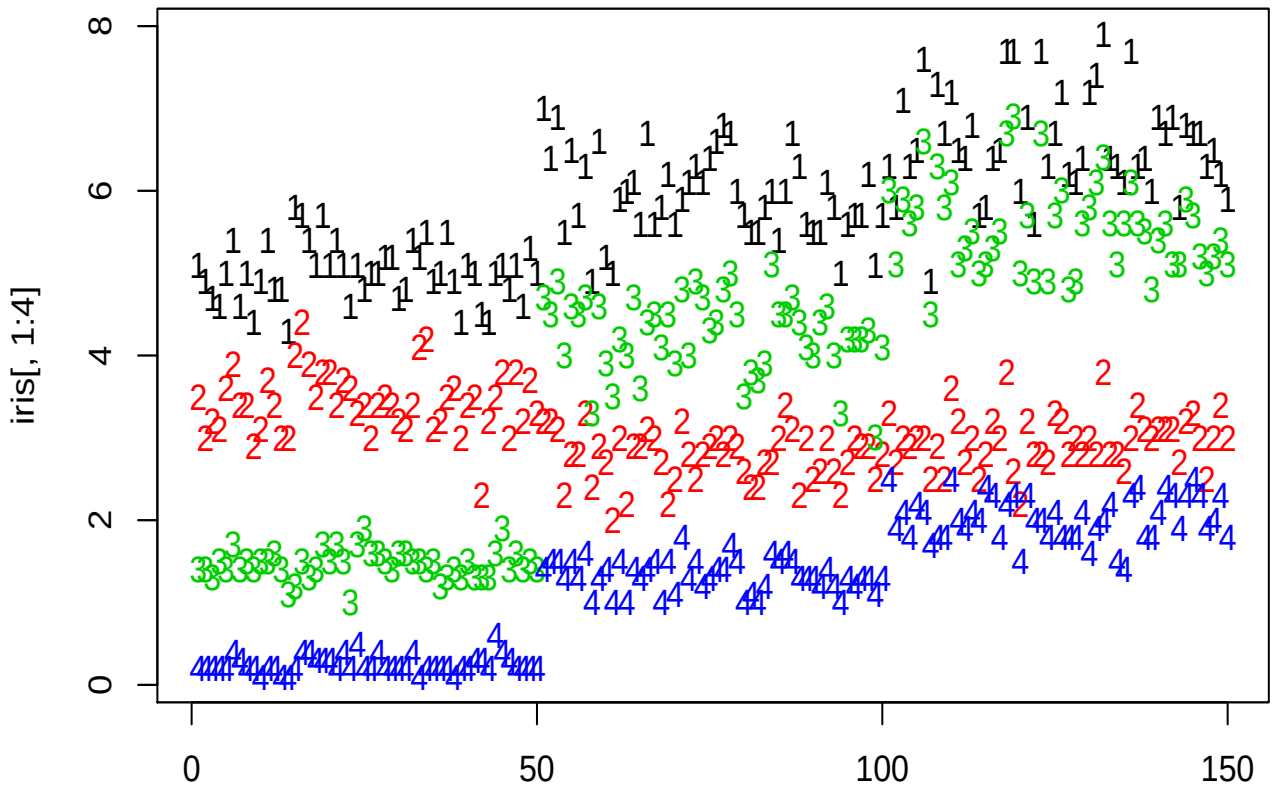
Plots of multiple variables

```
# scatter plot matrix (data frame)
pairs(iris[ , 1:4])
```



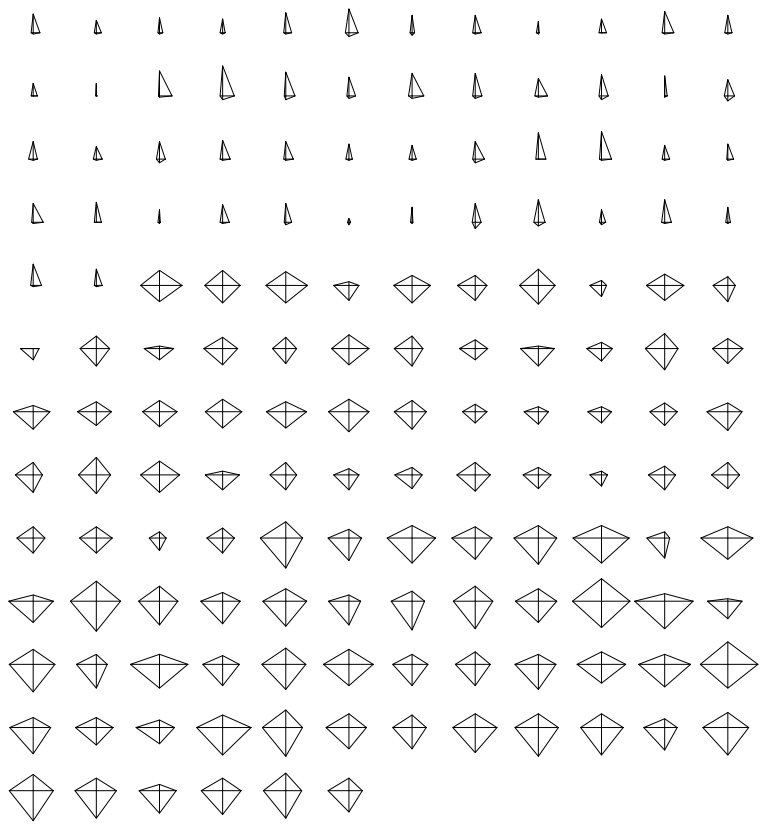
Plots of multiple variables

```
# scatter plot matrix (data frame)
matplot(iris[ , 1:4])
```



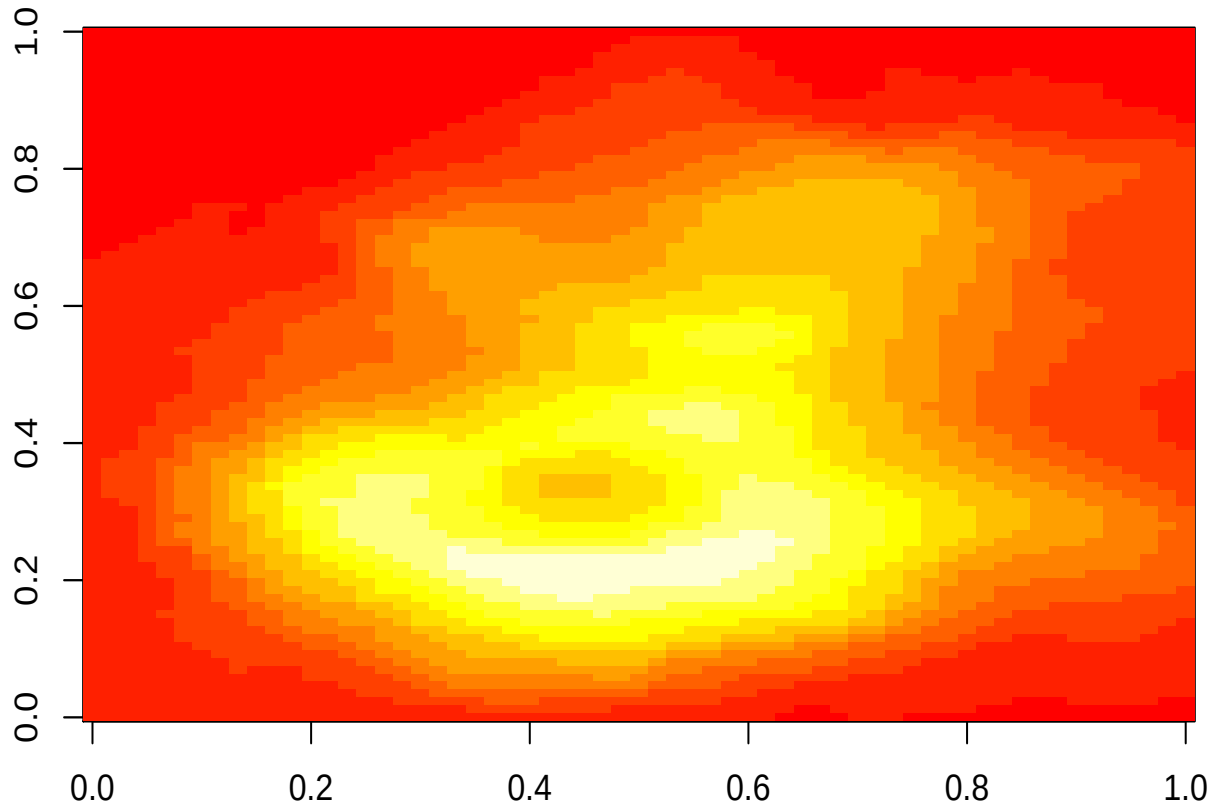
Plots of multiple variables

```r
# star plot (data frame)
stars(iris[ , 1:4])
```



Plots of multiple variables

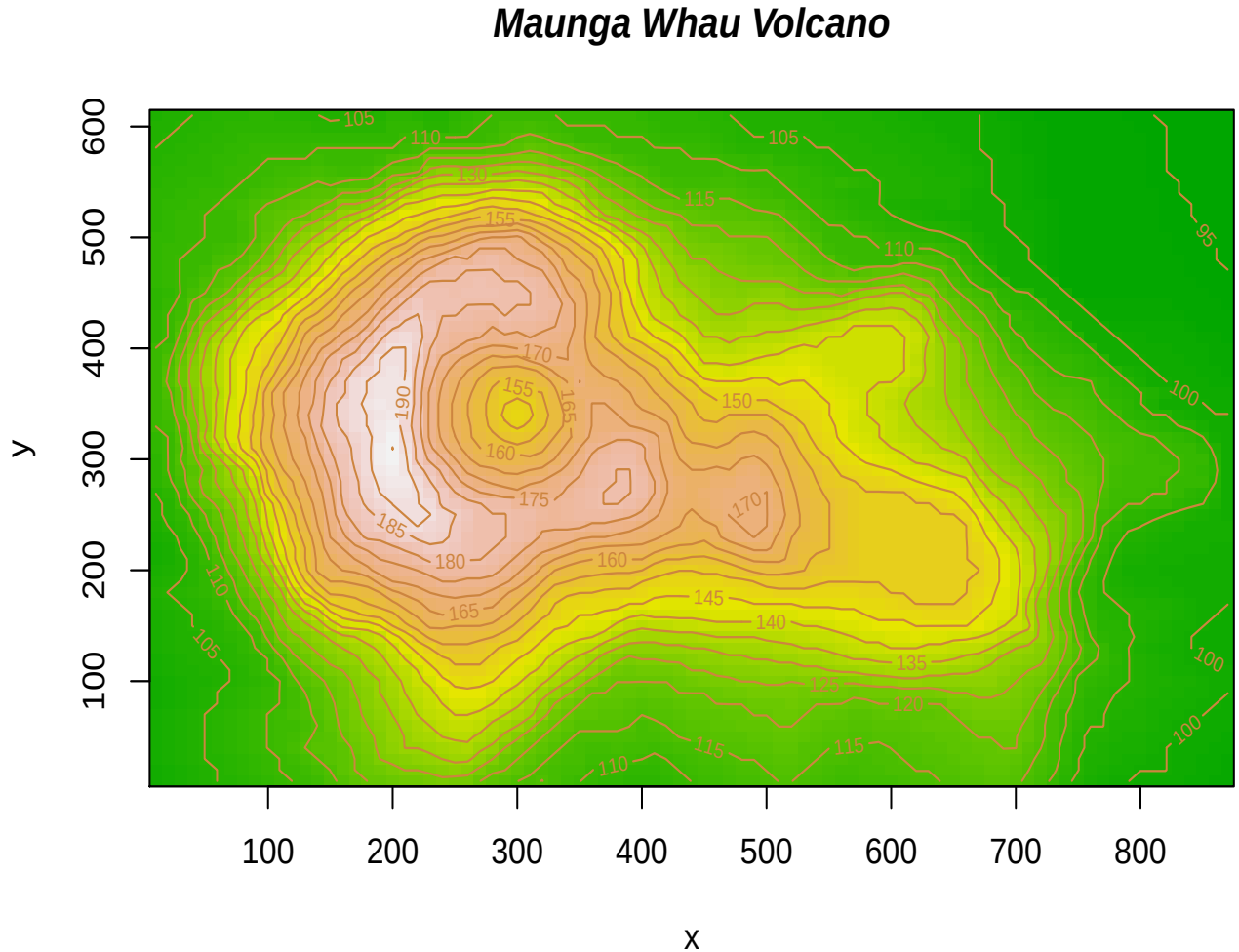
```
# color image (matrix)
image(t(volcano)[ncol(volcano):1, ])
```



Plots of multiple variables

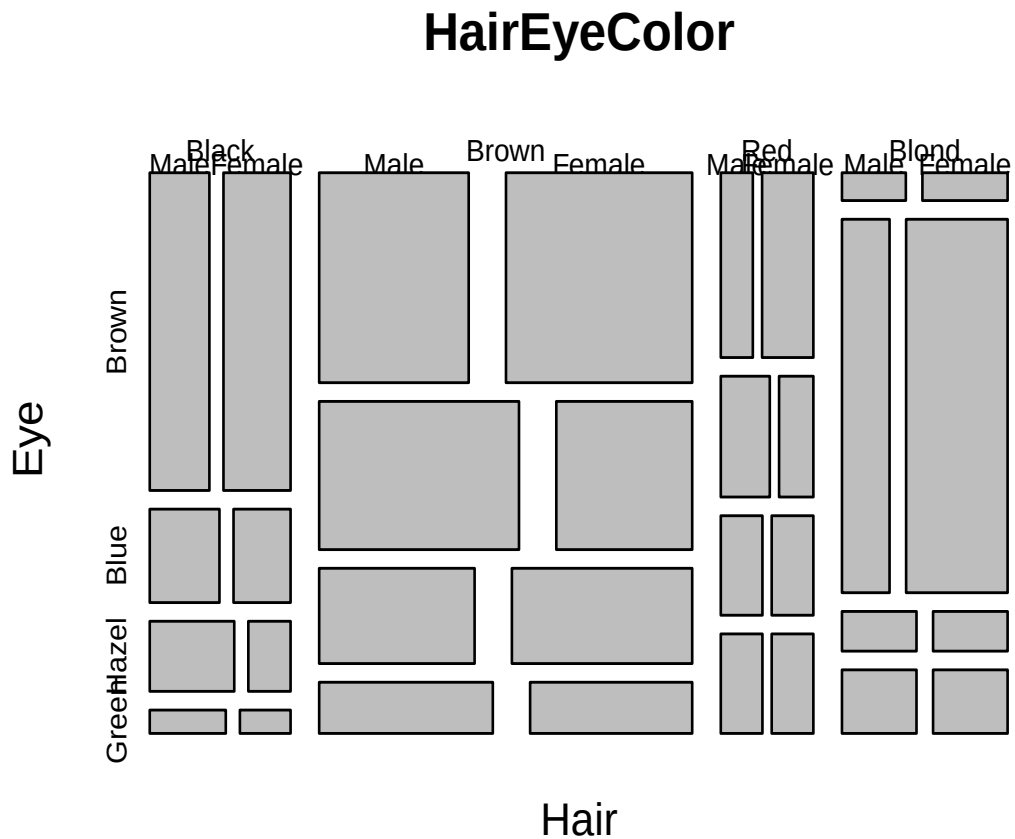
```
# display of Maunga Whau volcano
x <- 10*(1:nrow(volcano))
y <- 10*(1:ncol(volcano))
image(x, y, volcano, col = terrain.colors(100), axes = FALSE)
contour(x, y, volcano, levels = seq(90, 200, by = 5),
        add = TRUE, col = "peru")
axis(1, at = seq(100, 800, by = 100))
axis(2, at = seq(100, 600, by = 100))
box()
title(main = "Maunga Whau Volcano", font.main = 4)
```

Plots of multiple variables



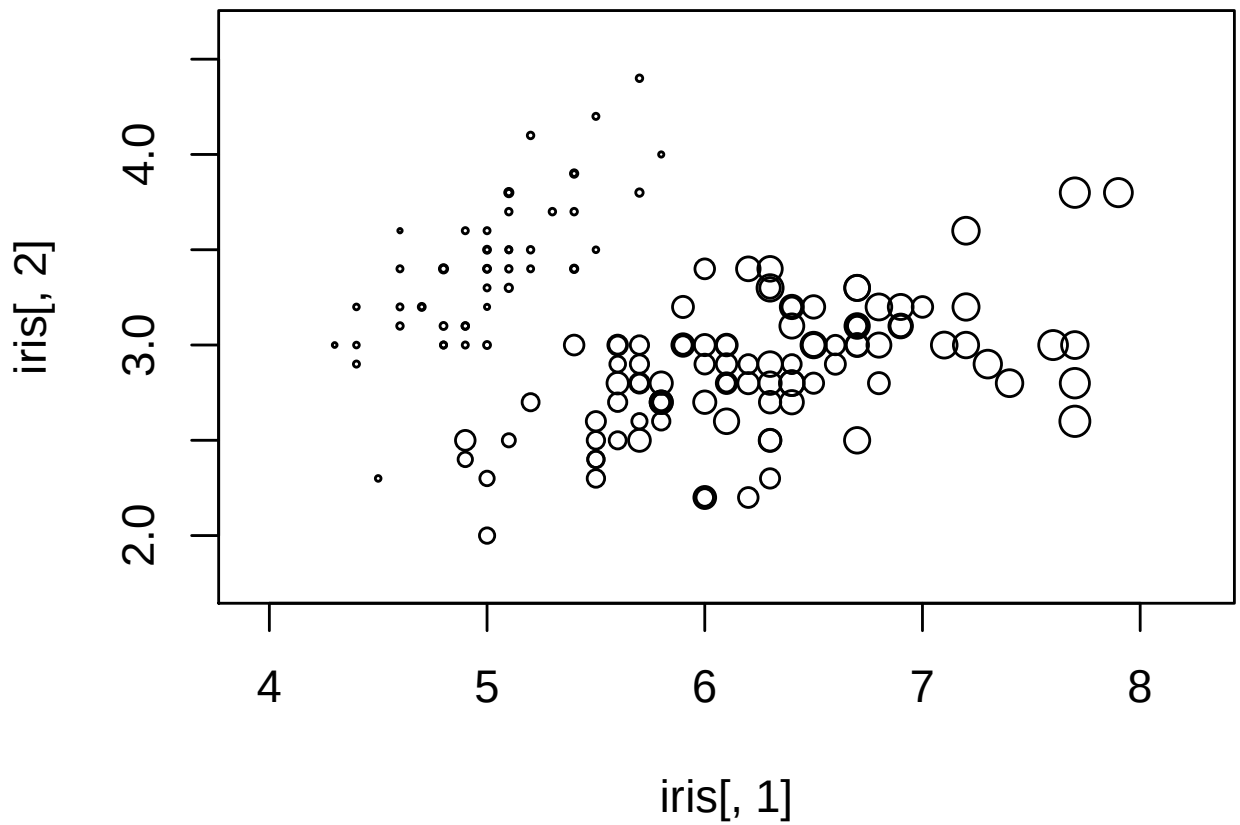
Plots of multiple variables

```
# mosaic plot of N-D tables
mosaicplot(HairEyeColor)
```



Plots of multiple variables

```r
# symbols scatter plots
symbols(iris[, 1], iris[, 2], circles = iris[, 3]/100,
        inches = FALSE)
```



Graphics Parameters

Graphics Functions and Arguments

- Plot functions usually come with various arguments
- Typically, the first argument(s) is the data object(s) to be plotted
- Most of the other arguments have default options
- Graphic arguments have a consisting naming convention, but there will always be some exception

Graphical Parameters

Graphical Arguments

- Some arguments are specific to a function (e.g. `horiz` or `beside` in `barplot()`)
- Other arguments are more general (e.g. `col`, `xlab`, `ylab`)
- General graphical parameters are listed in the documentation of the function `par()`
- See `?par` for more information

How to choose a graphics approach?

- look first for an existing function that does what you want —or something similar to what you want (don't reivent the wheel!)
- Existing plotting functions can be combined and customized by using optional arguments or graphical parameters
- For exploratory data analysis (quick and dirty) the plotting functions in `"graphics"` is a good option