Base Graphics (part 1)

Graphics

Gaston Sanchez

R Coding Compendium

Donate

About

In this slides we cover the base graphics system: `"graphics"` packages

R Graphics

2 main graphics systems supported by their corresponding packages:

`"graphics"` & `"grid"`

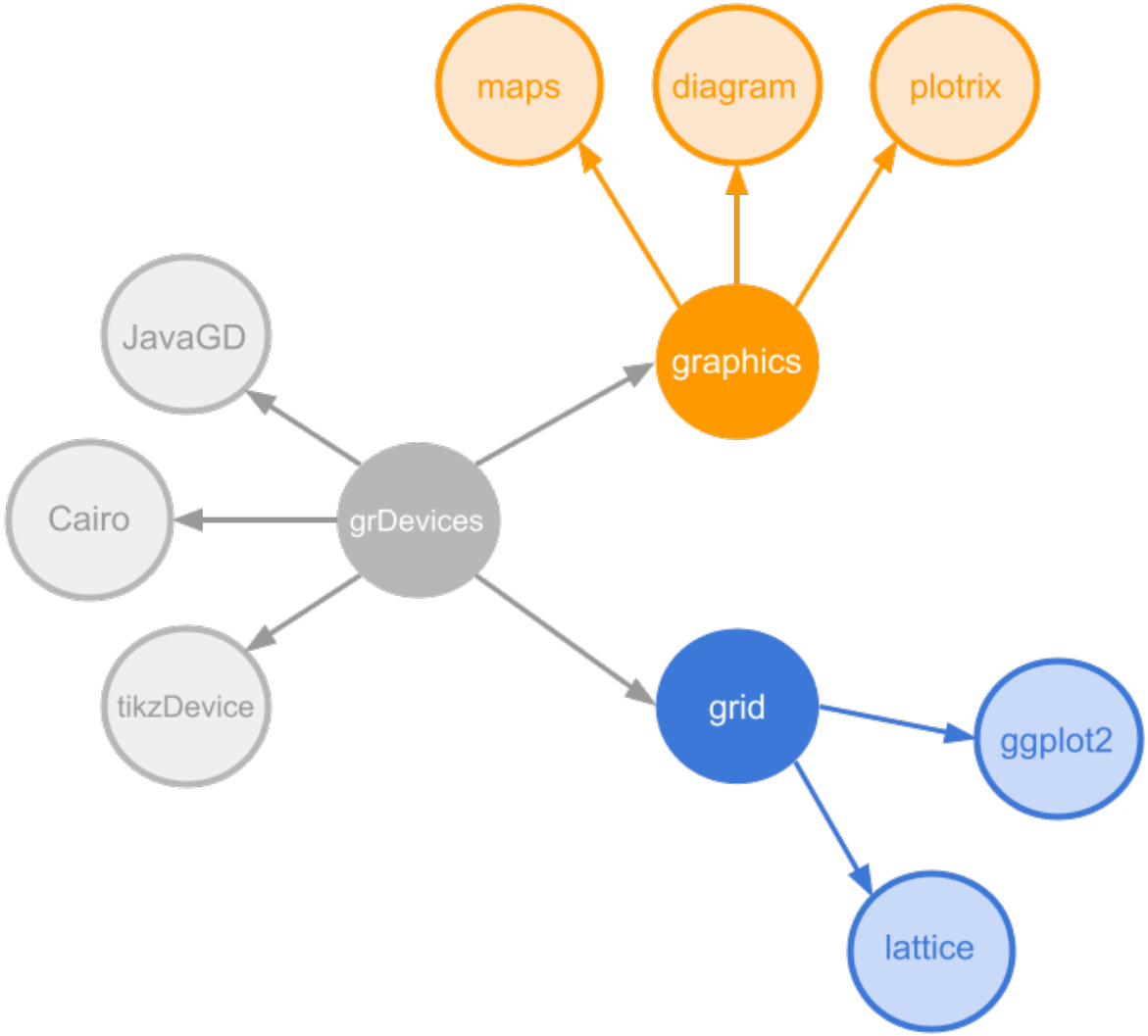
Graphics Systems in R

Simply put:

- `"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

Basics of Graphics in R

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.

The "grid" package does not provide functions for drawing complete plots. In other words, "grid" is not used directly to produce statistical plots. Instead, it is used to build other graphics packages like "lattice" or "ggplot2".

As you may know, "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 Base Graphics

Traditional (Base) Graphics

Graphics functions can be divided into two main types:

- **high-level** functions produce complete plots, for example
  - `barplot()`
  - `hist()`
  - `boxplot()`
  - `dotchart()`
- **low-level** functions add further output to an existing plot
  - `text()`
  - `points()`
  - `lines()`
  - `legend()`
  - *etc*

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

The `plot()` function

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

- one single variable
- two variables
- multiple variables

One variable graphics

Plots of One Variable

High-level graphics of a single variable

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

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

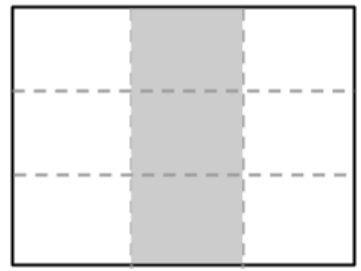
One variable objects

Vector / Factor

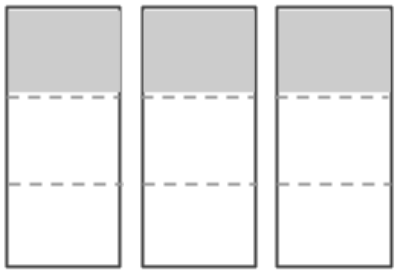
1-D table

row (matrix)

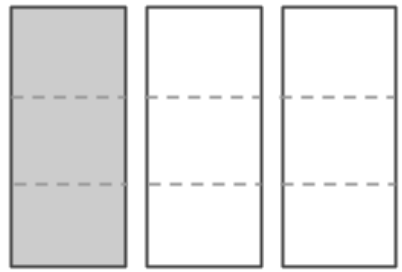
column (matrix)



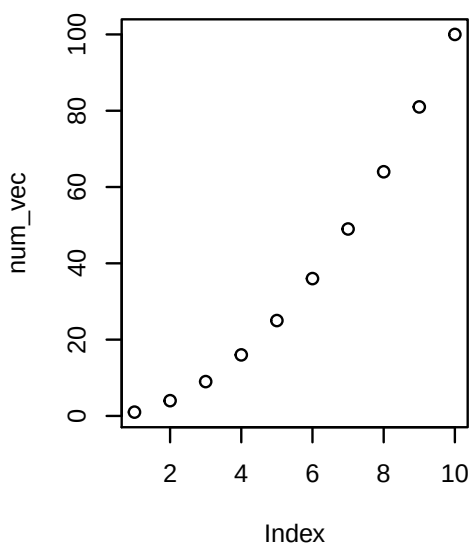
row (data.frame)

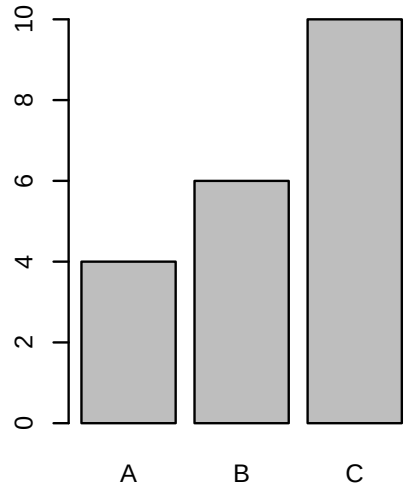


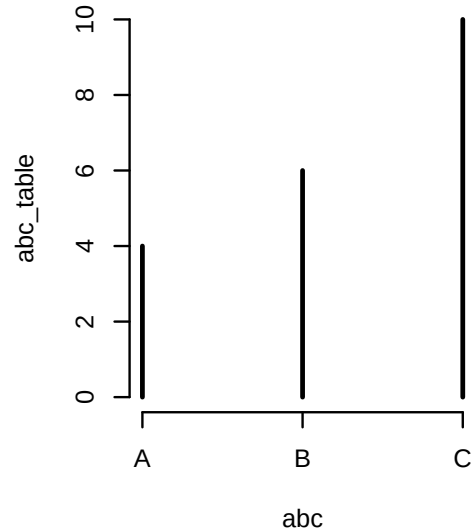
column (data.frame)



```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)
```


num_vec
0
20
40
60
80
100
2
4
6
8
10
Index


0
2
4
6
8
10
A
B
C

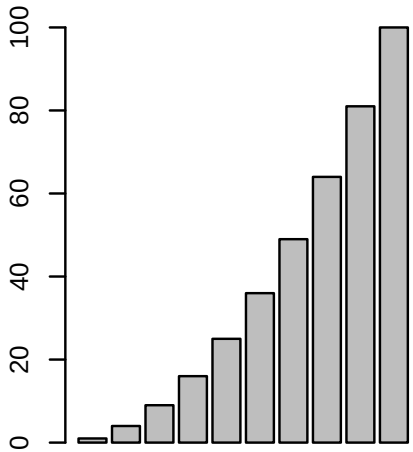

abc_table
0
2
4
6
8
10
A
B
C
abc

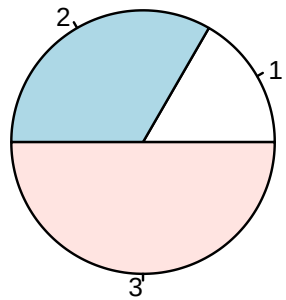
More high-level graphics of a single variable

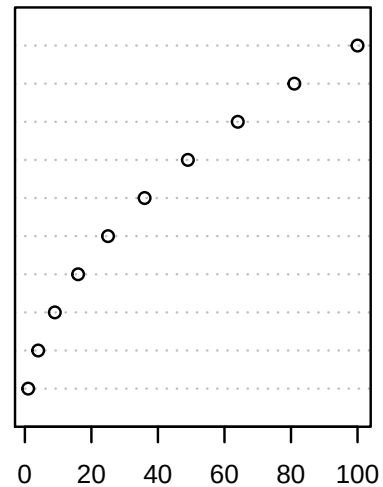
Function	Data	Function
`barplot()`	numeric	barchart
`pie()`	numeric	piechart
`dotchart()`	numeric	dotplot
`boxplot()`	numeric	boxplot
`hist()`	numeric	histogram
`stripchart()`	numeric	1-D scatterplot
`stem()`	numeric	stem-and-leaf plot

Examples: one signle variable plots

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

# pie chart
pie(1:3)

# dot plot
dotchart(num_vec)
```

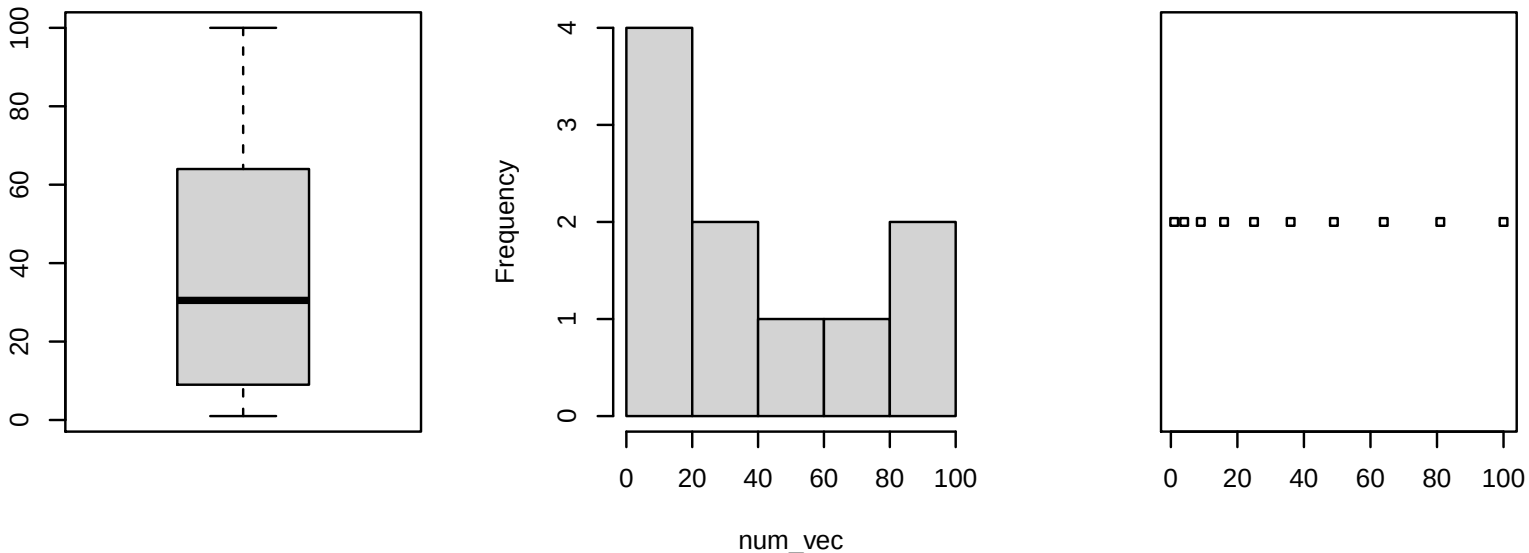

0
20
40
60
80
100


2
1
3


0
20
40
60
80
100

Examples: one signle variable plots

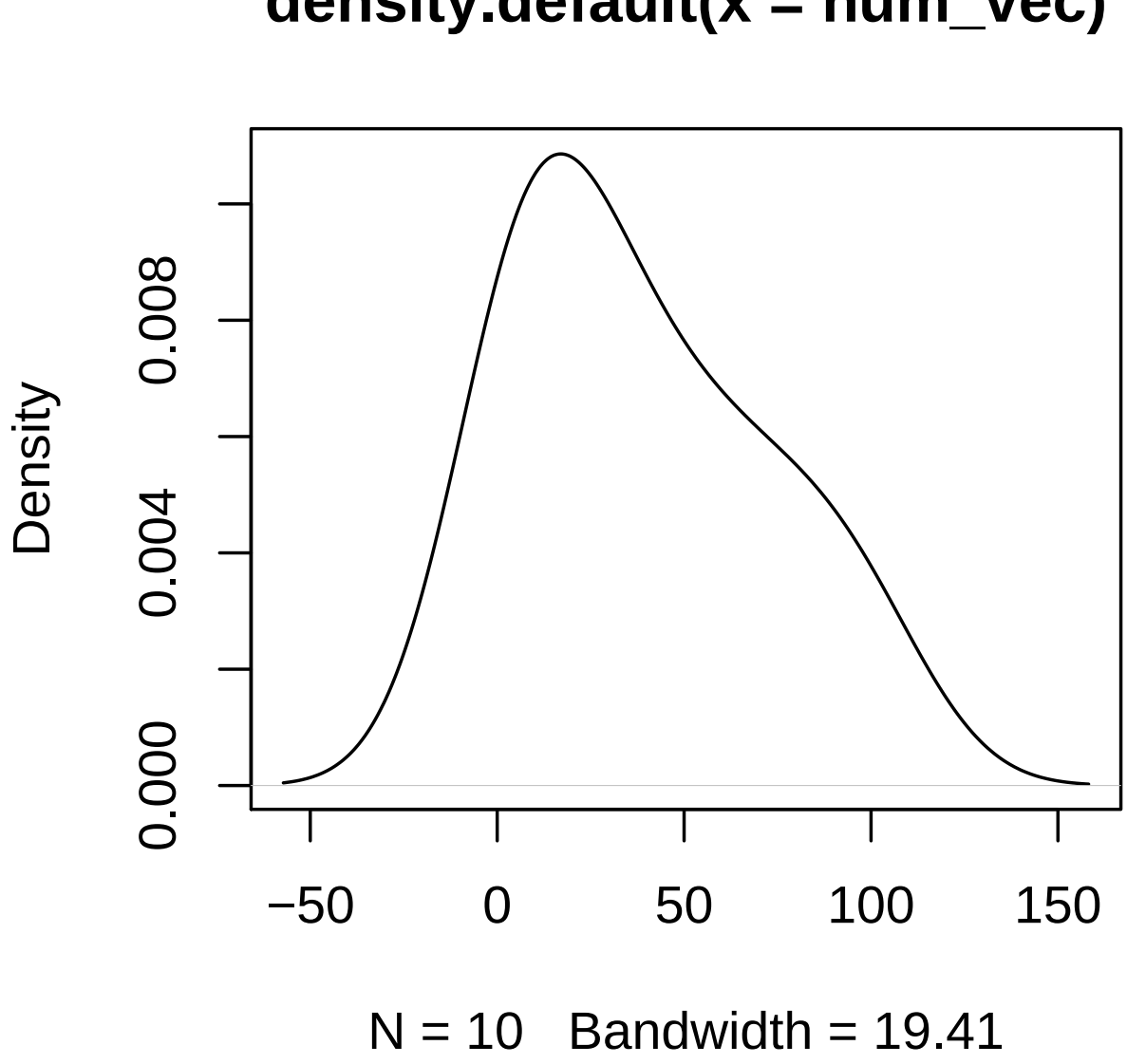
```r
# barplot numeric vector
boxplot(num_vec)

# pie chart
hist(num_vec)

# dot plot
stripchart(num_vec)
```


Histogram of num_vec
Frequency
num_vec
0
20
40
60
80
100
0
1
2
3
4

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.

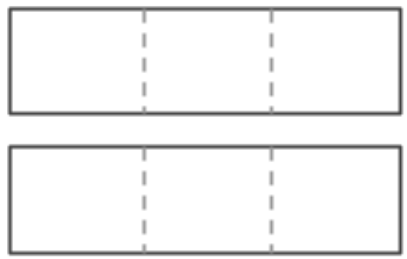
```
# kernel density curve
dens <- density(num_vec)
plot(dens)
```


density.default(x = num_vec)
Density
0.000
0.004
0.008
−50
0
50
100
150
N = 10 Bandwidth = 19.41

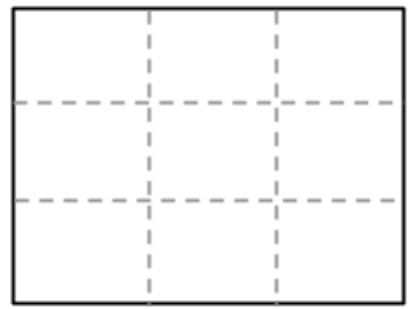
Plots of Two Variables

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

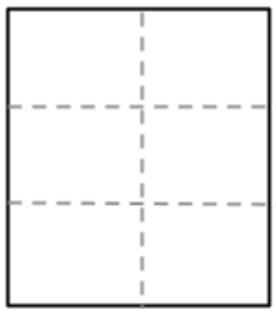
2 numeric vectors
num vector, factor
factor, num vector
2 factors



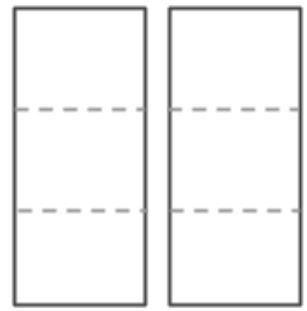
2-D table
(frequency or
crosstable)



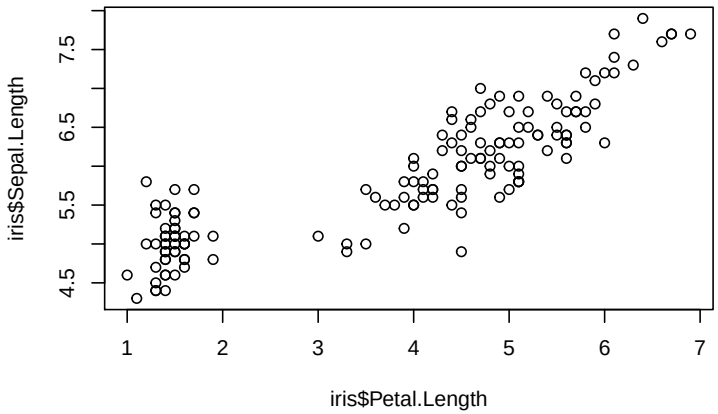
2-column
(numeric matrix)

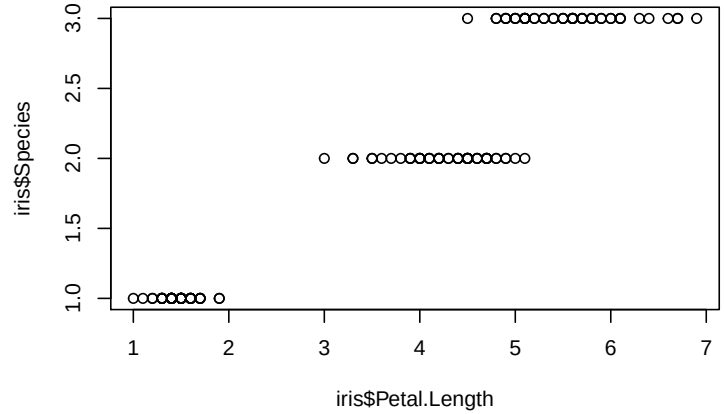


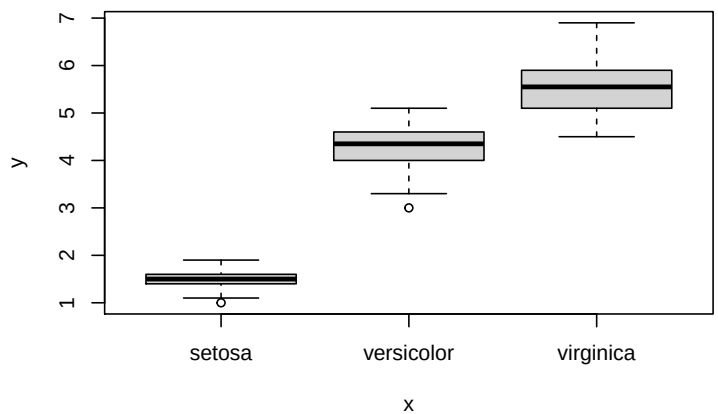
2-column
(numeric data.frame)

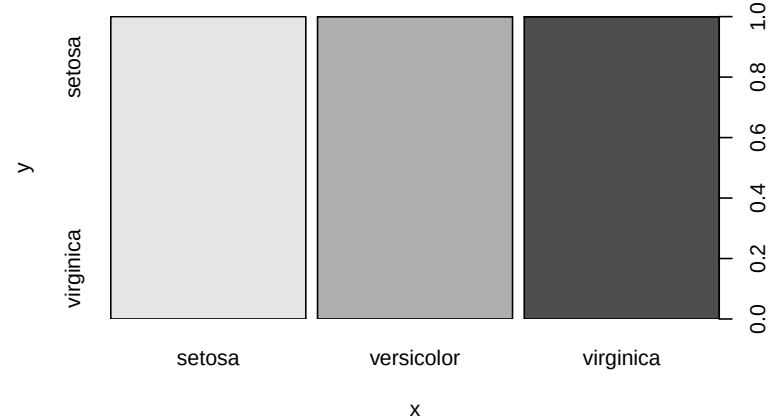


```r
# plot numeric, numeric
plot(iris$Petal.Length, iris$Sepal.Length)

# 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)
```


iris$Sepal.Length
iris$Petal.Length
4.5
5.5
6.5
7.5
1
2
3
4
5
6
7


iris$Species
iris$Petal.Length
1.0
1.5
2.0
2.5
3.0
1
2
3
4
5
6
7

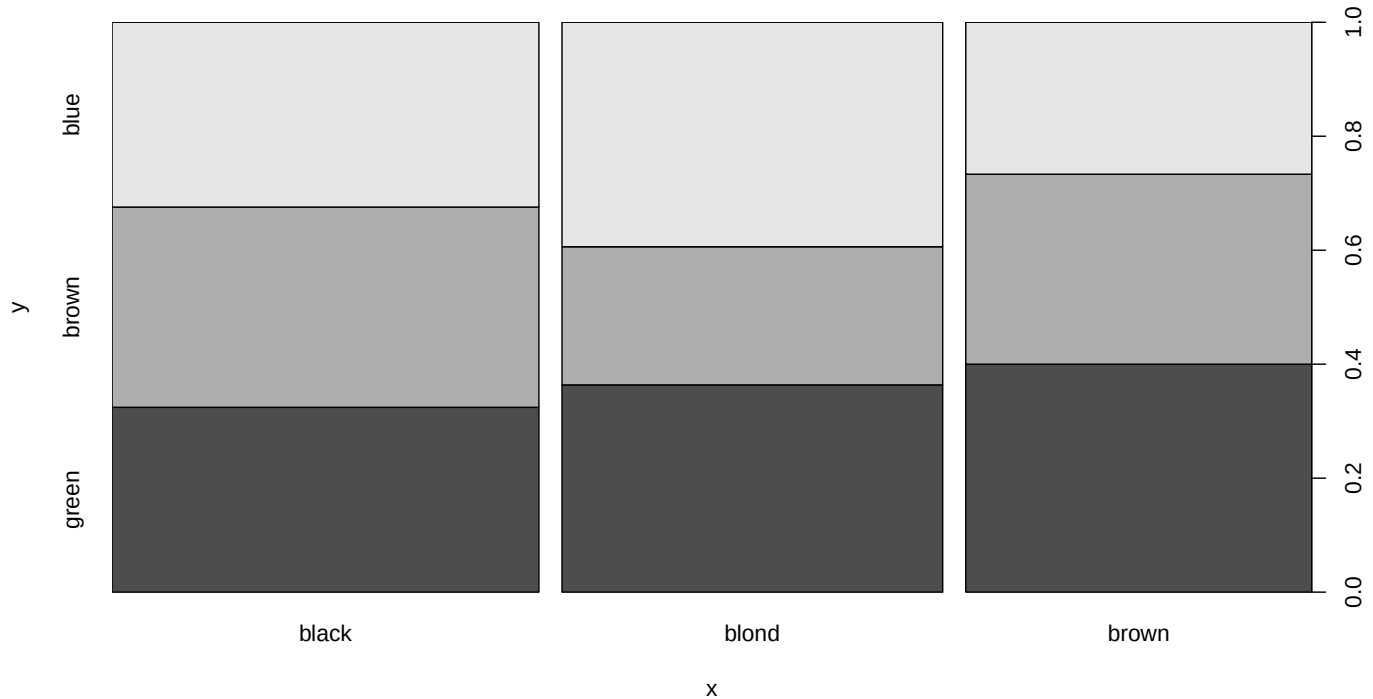

y
x
setosa
versicolor
virginica
1
2
3
4
5
6
7


y
x
setosa
virginica
setosa
versicolor
virginica
0.0
0.2
0.4
0.6
0.8
1.0

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))
```

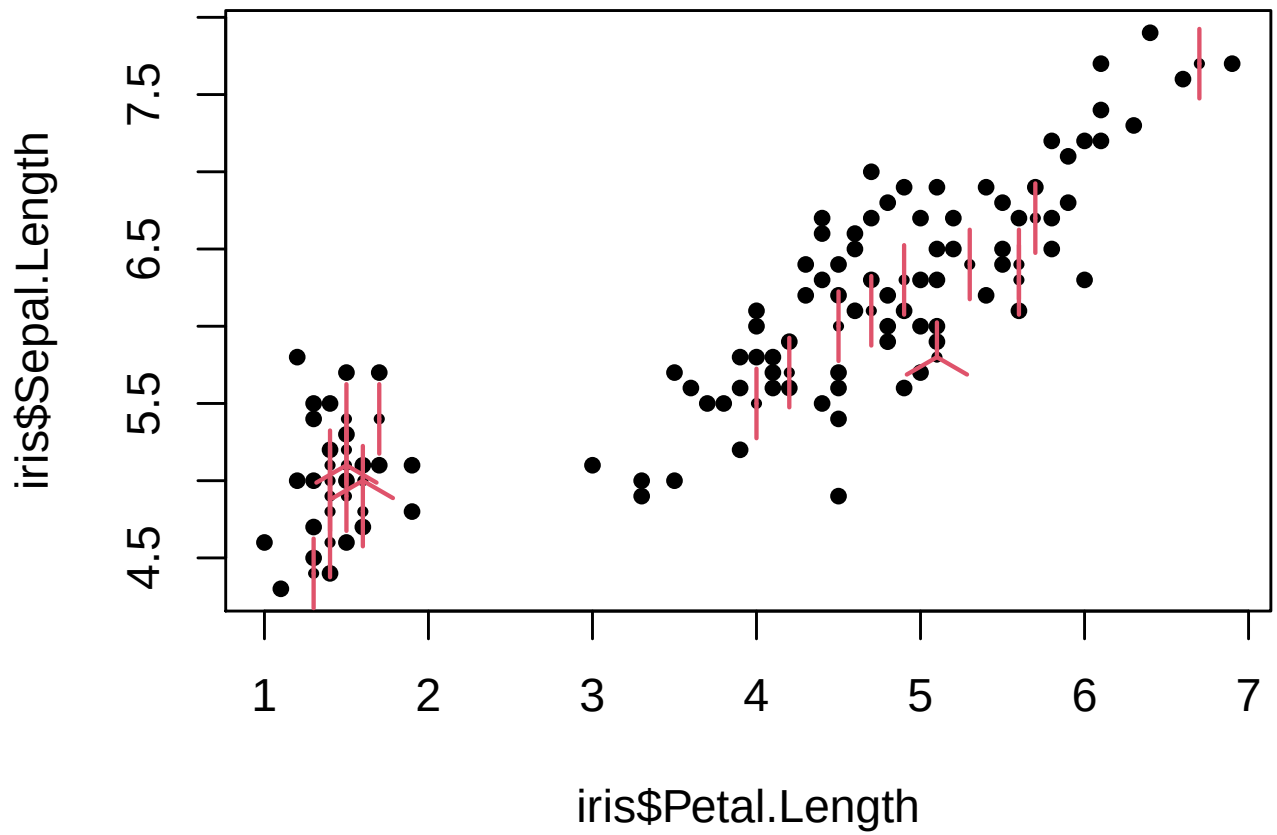
```r
# plot factor, factor
plot(hair, eye)
```



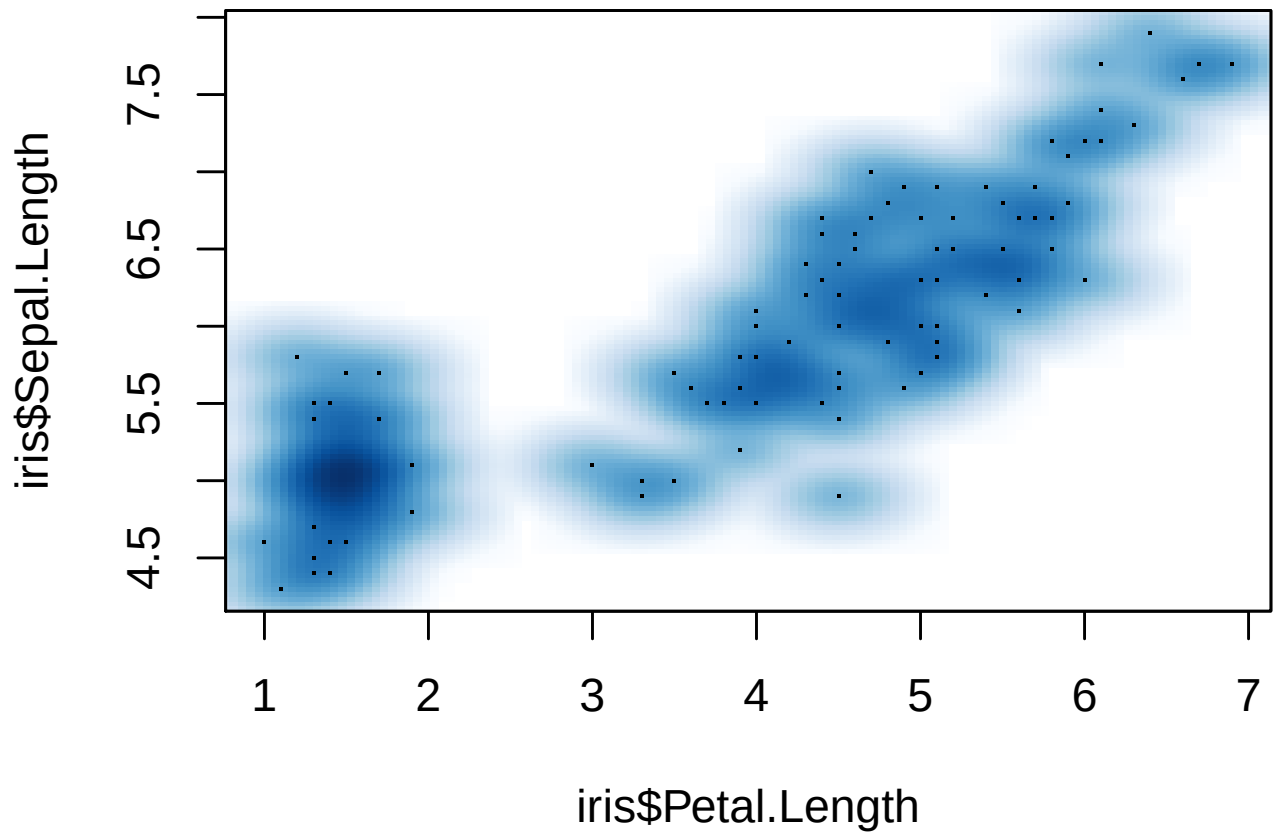
More high-level graphics of two variables

Function	Data	Function
`sunflowerplot()`	numeric, numeric	sunflower scatterplot
`smoothScatter()`	numeric, numeric	smooth scatterplot
`boxplot()`	list of numeric	boxplots
`barplot()`	matrix	stacked 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	mosaicplot

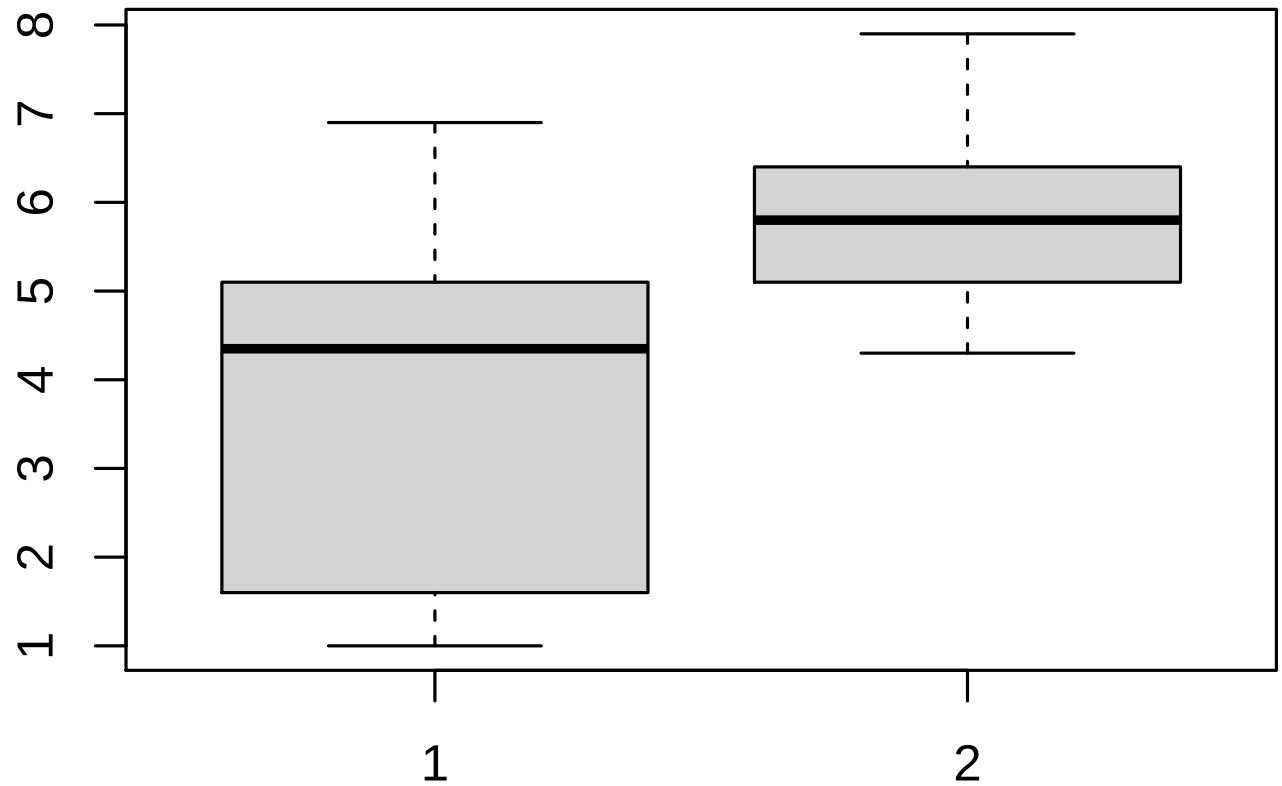
```r
# sunflower plot (numeric, numeric)
sunflowerplot(iris$Petal.Length, iris$Sepal.Length)
```



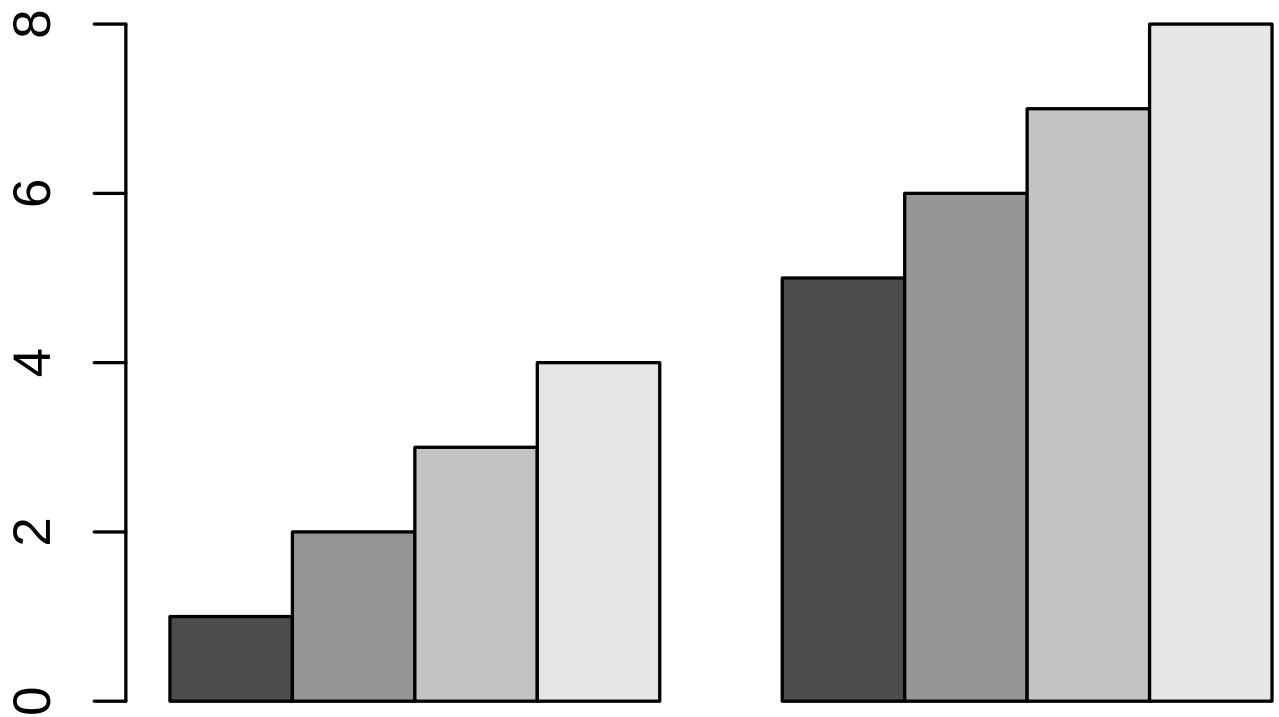
```r
# smooth scatter plot (numeric, numeric)
smoothScatter(iris$Petal.Length, iris$Sepal.Length)
```



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



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



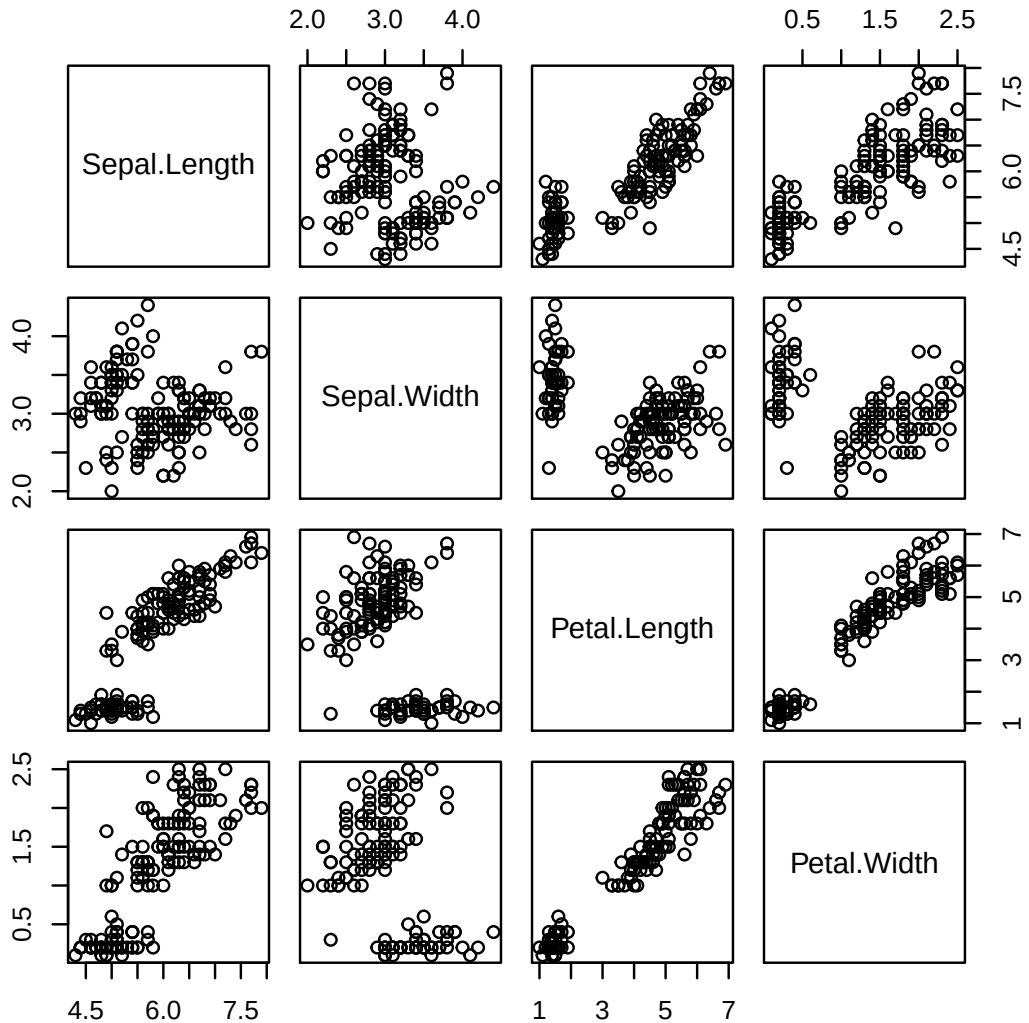
```r
# conditional density plot (numeric, factor)
cdplot(iris$Petal.Length, iris$Species)
```

Plots of Two Variables

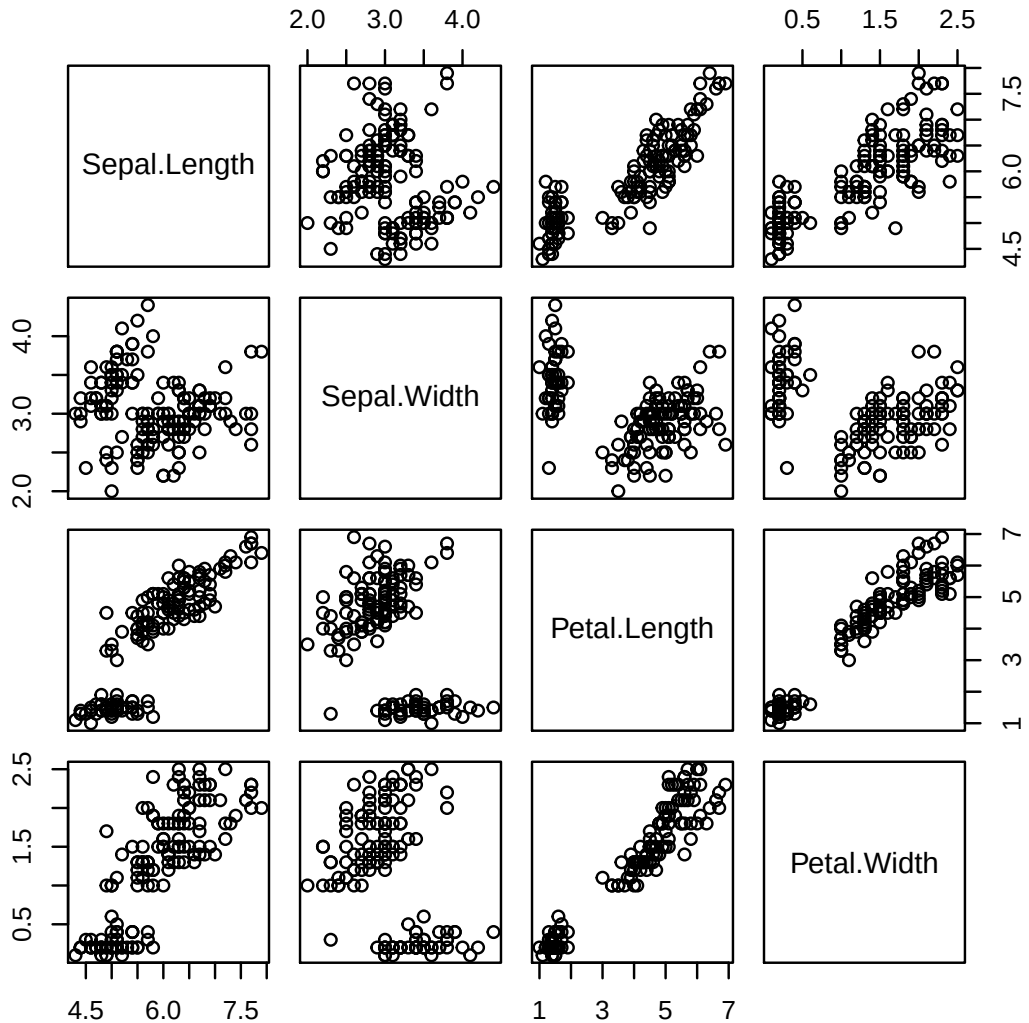
More high-level graphics of two variables

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

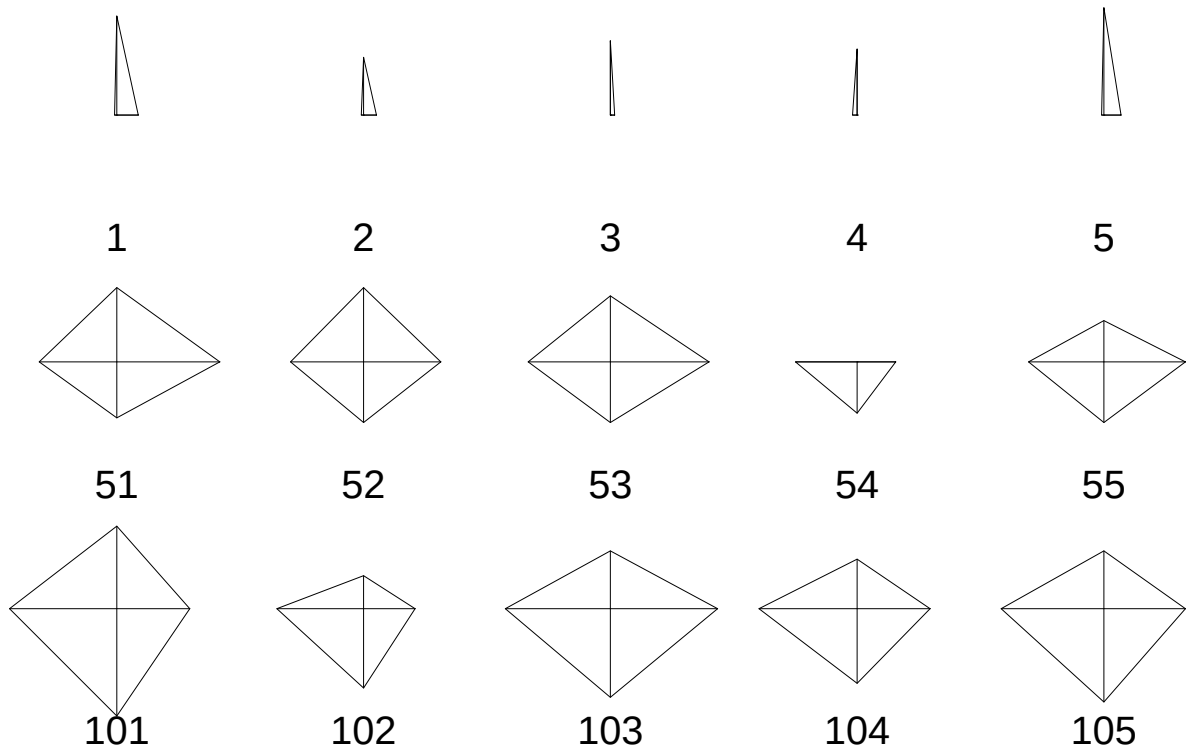
```r
# scatter plot matrix (data frame)
plot(iris[ , 1:4])
```



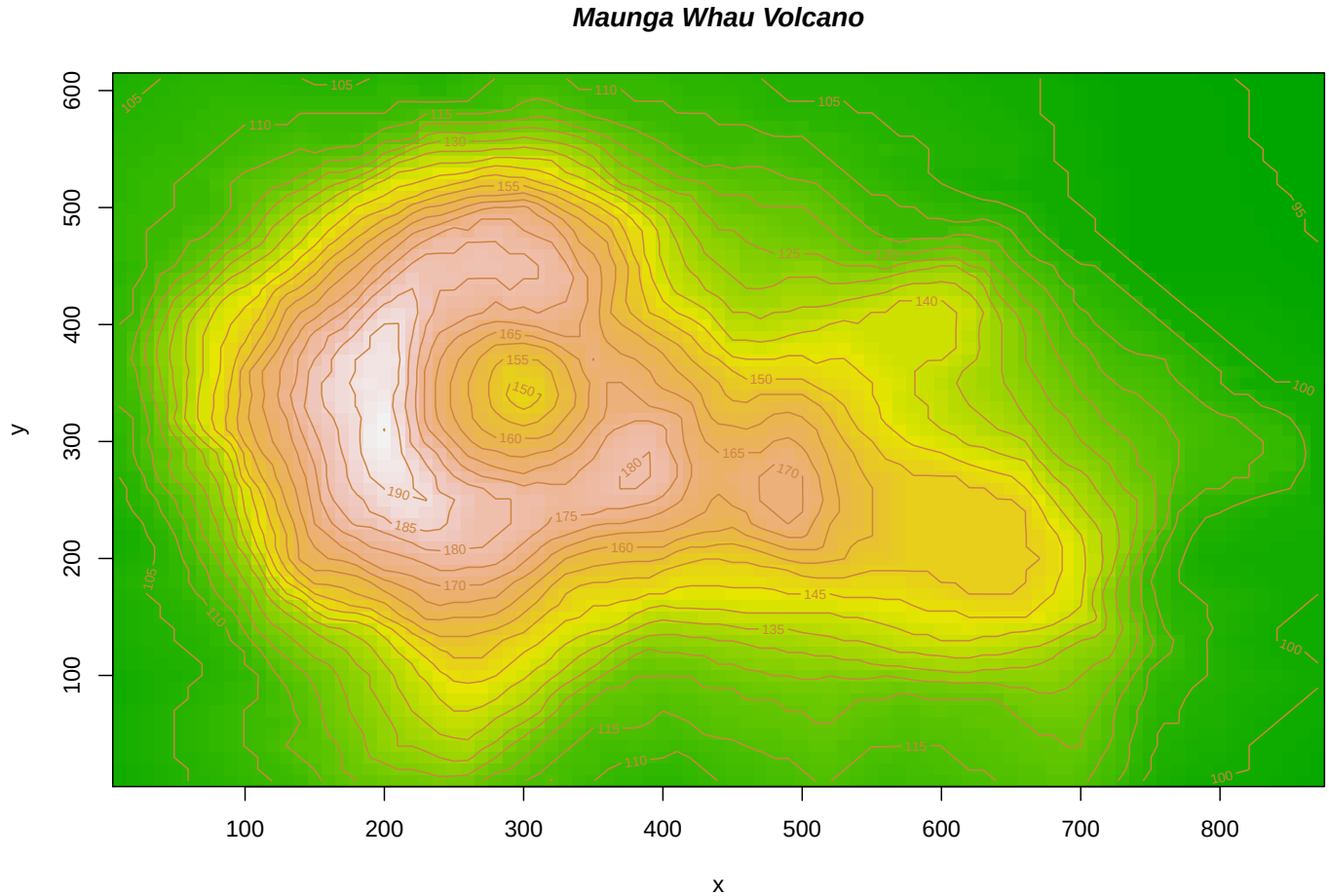
```r
# scatter plot matrix (data frame)
pairs(iris[ , 1:4])
```



```r
# star plot (data frame)
stars(iris[c(1:5,51:55,101:105), 1:4], nrow = 3)
```



```r
# 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)
```


Maunga Whau Volcano
y
x
100
200
300
400
500
600
700
800

Graphics Parameters

- 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

Graphics Parameters

- 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

Graphics Parameters

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
- For more reporting-quality graphics, `"ggplot2"` may be the "go to" option.

Donation

If you find any value and usefulness in this set of slides, please consider making a one-time donation in any amount (via paypal). Your support really matters.

Donate

https://www.paypal.com/donate?business=ZF6U7K5MW25W2&currency_code=USD