Modelling in Physical Geography

Excercise 2

Konsta Happonen

Plotting

When doing analysis, it is crucial to get to know one's data. The fastest way to do this is "eyeballing", or visual inspection. Graphical representations of the data can be used to quickly gain information about the variation within and between variables. For this excercise we will be using one of the R default datasets called `brambles`, which contains the locations and age classes of raspberry stems in a square. For more information about the dataset, type `help(brambles)` on the command line.

```r
data(brambles)
str(brambles)
```

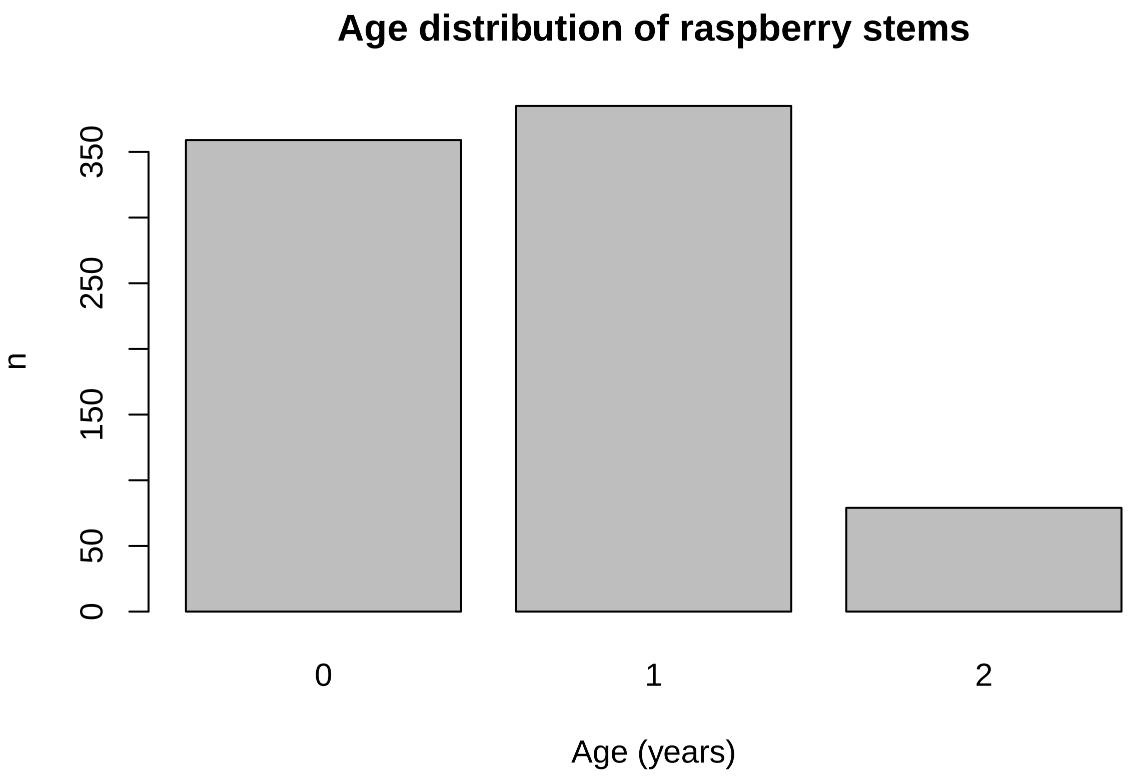
```
## 'data.frame':    823 obs. of  3 variables:
##  $ x  : num  0.677 0.676 0.681 0.683 0.776 0.794 0.944 0.948 0.983 0.986 ...
##  $ y  : num  0.001 0.022 0.031 0.038 0.028 0.033 0.011 0.01 0.077 0.084 ...
##  $ age: num  0 0 0 0 0 0 0 0 0 0 ...
```

One discrete variable

Barplots

To see the distribution of one discrete variable, we can use bar plots. We can treat the `brambles` age as a discrete variable, and plot it. The (horizontal) x-axis of the resulting plot lists all the unique ages in our data, and the (vertical) y-axis represents their counts. The arguments `main`, `xlab` and `ylab` control the plot title and axis labels. Use them to make your graph more informative.

```r
ages <- table(brambles$age) # Summarize the frequency of each age class
barplot(ages,
        main = "Age distribution of raspberry stems",
        xlab = "Age (years)",
        ylab = "n") # Plot them.
```



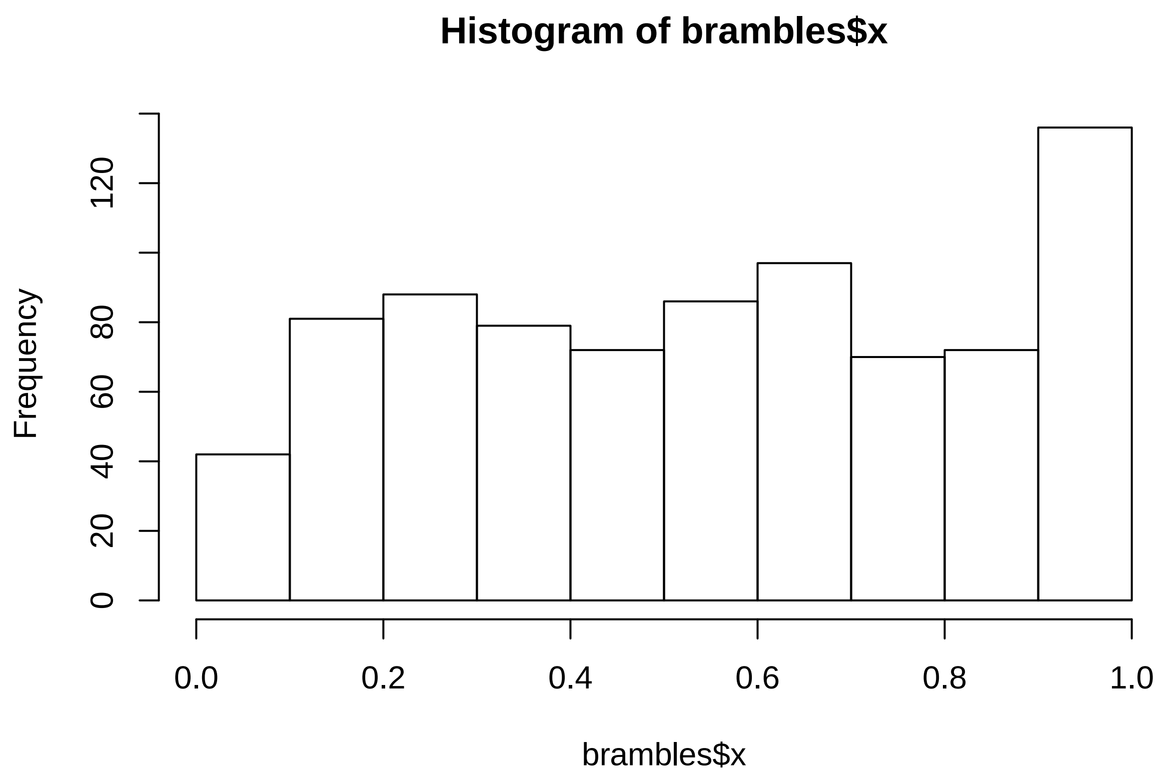
One continuous variable

The distribution of a continuous variable is most easily visualized by histograms and boxplots.

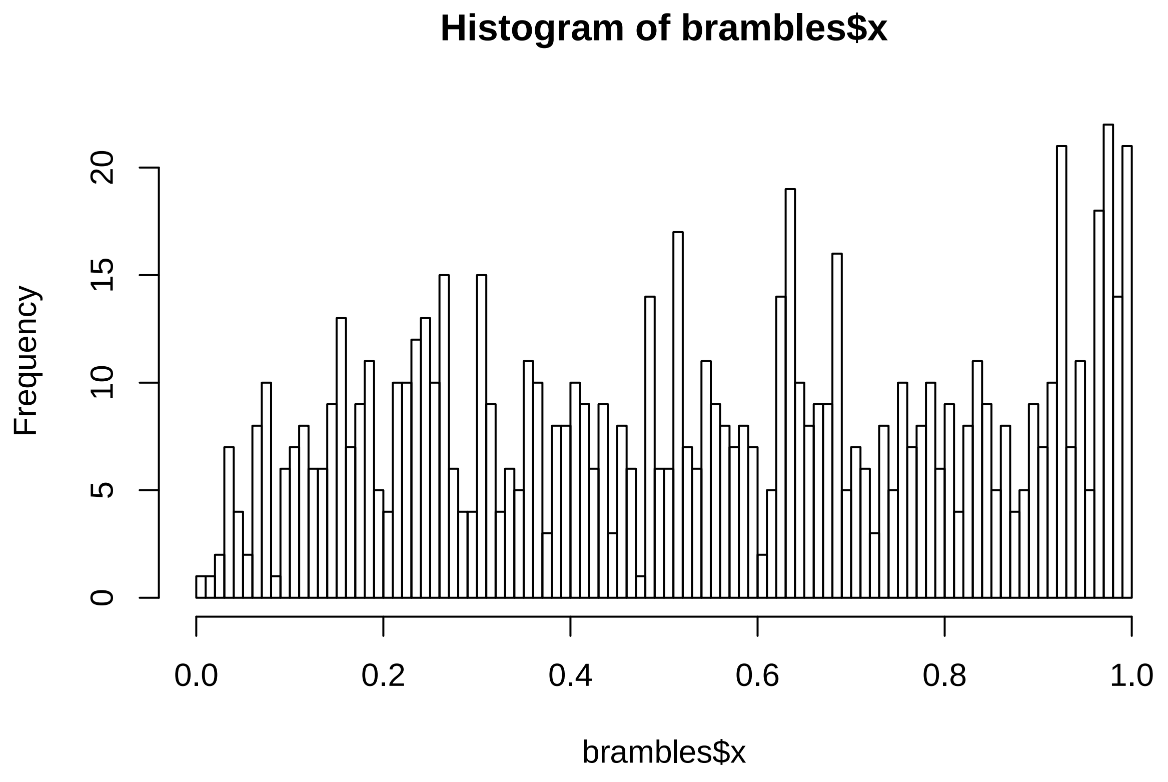
Histograms

A histogram discretizes the continuous variable to discrete classes, and then represents their frequencies with rectangles. The area of each rectangle is directly proportional to the frequency of that class. The argument `breaks` can be used to control the coarseness of the discretisation.

```
hist(brambles$x) # plot the distribution of raspberries along the x-axis
```



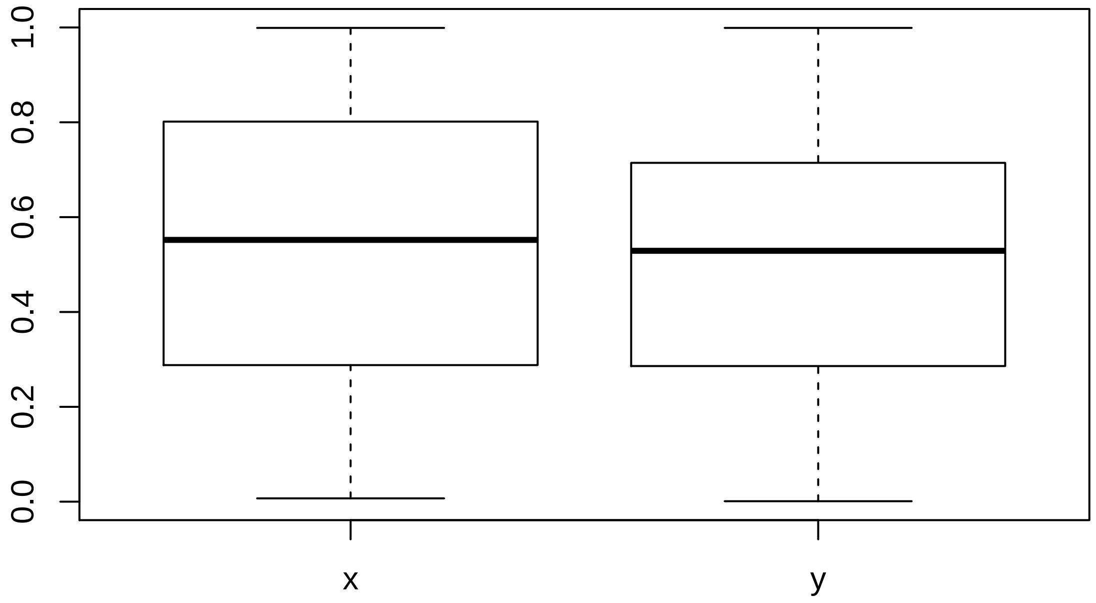
```
hist(brambles$x, breaks=100)
```



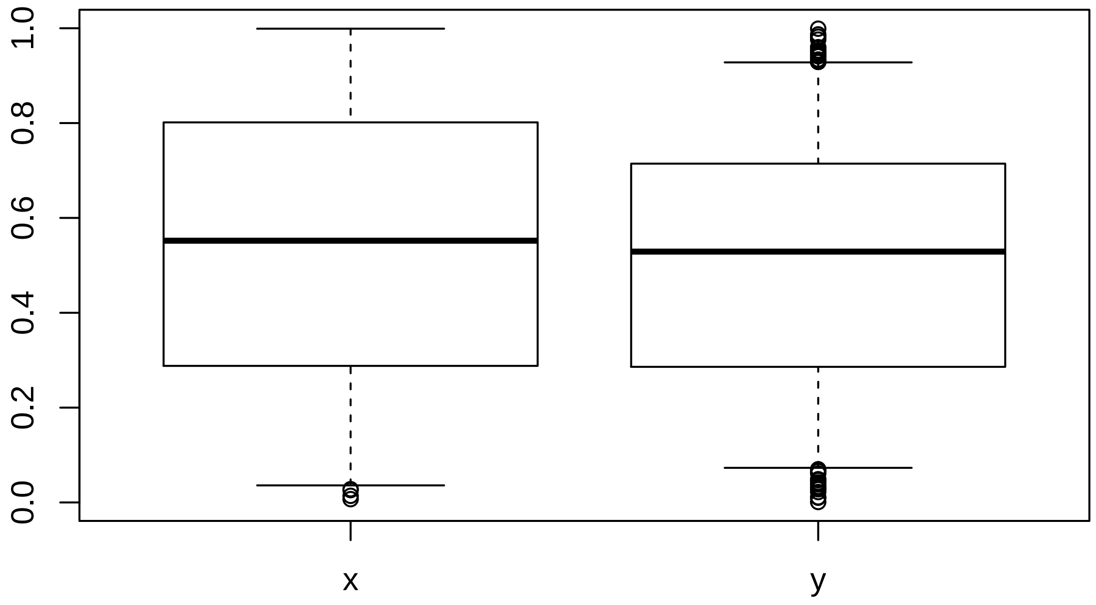
Boxplots

A box plot visualizes the *quartiles* of a set of numbers. Quartiles (Q1,Q2,Q3) are calculated by first arranging the values according to their value. Q2, also called the *median*, is the middle number between the minimum and maximum values. Accordingly, Q1 is the middle number between the minimum and Q2, and Q3 is the middle number between Q2 and the maximum. The boxplot displays the minimum, Q1, Q2,Q3, and the maximum. In addition, any data points more than 1.5 interquartile ranges (distance between Q1 and Q3) away from the nearest quantile are displayed as separate points. The definition of outliers can be tweaked with the argument `range`. Boxplots are especially useful for identifying outliers (abnormally extreme values, often indicative of some kind of error in data collection) and comparing multiple variables on the same scale.

```r
boxplot(brambles[,c("x","y")])
```



```r
boxplot(brambles[,c("x","y")], range=0.5)
```



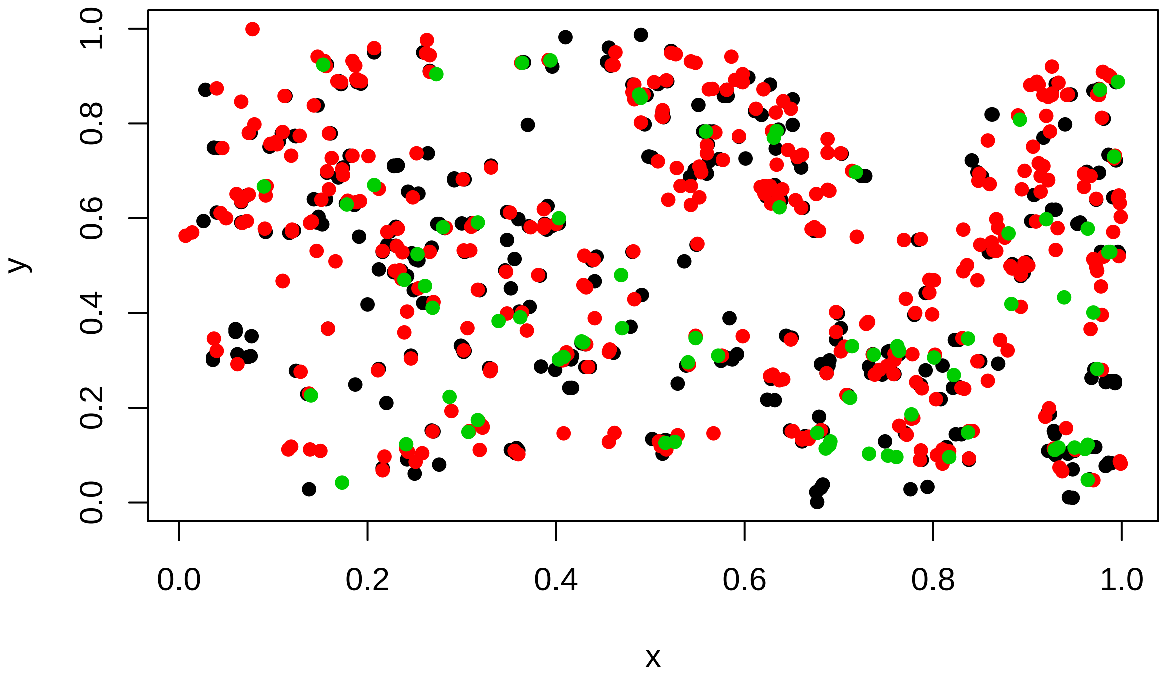
Two continuous variables

Often what is truly interesting in the relationship between our variables. To visualise these we can use scatterplots and lineplots

Scatterplots

In scatterplots, the values of two continuous values are mapped on the x- and y-axes. Each data point is represented a point in the plot. If the two variables are a “response” and an “explanatory” variable, the convention is to map the response on the y-axis. The argument `pch` can be used to change the point symbol, and the argument `col` to change the color of the points.

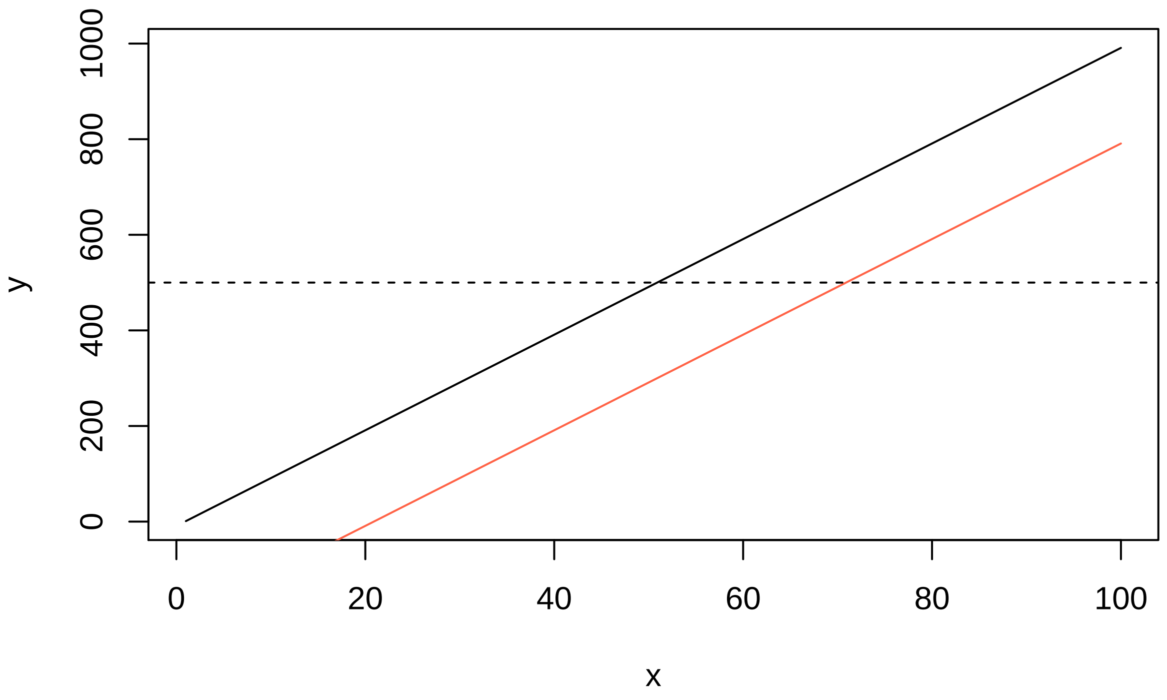
```
## Plot the spatia distribution of data with filled circles, colored by age
plot(y~x, data = brambles, col = factor(brambles$age), pch = 16)
```



Lineplots

Lineplots are good for displaying trends. The `brambles` data is ill-suited for dempnstrating this, so we will create some trends. We will use the command `seq()` to create regular sequences of data. We specify `type = "l"` in the ploting command to create a line plot. We can then use `lines()` to add another line in the plot. We can also use `abline()` to create

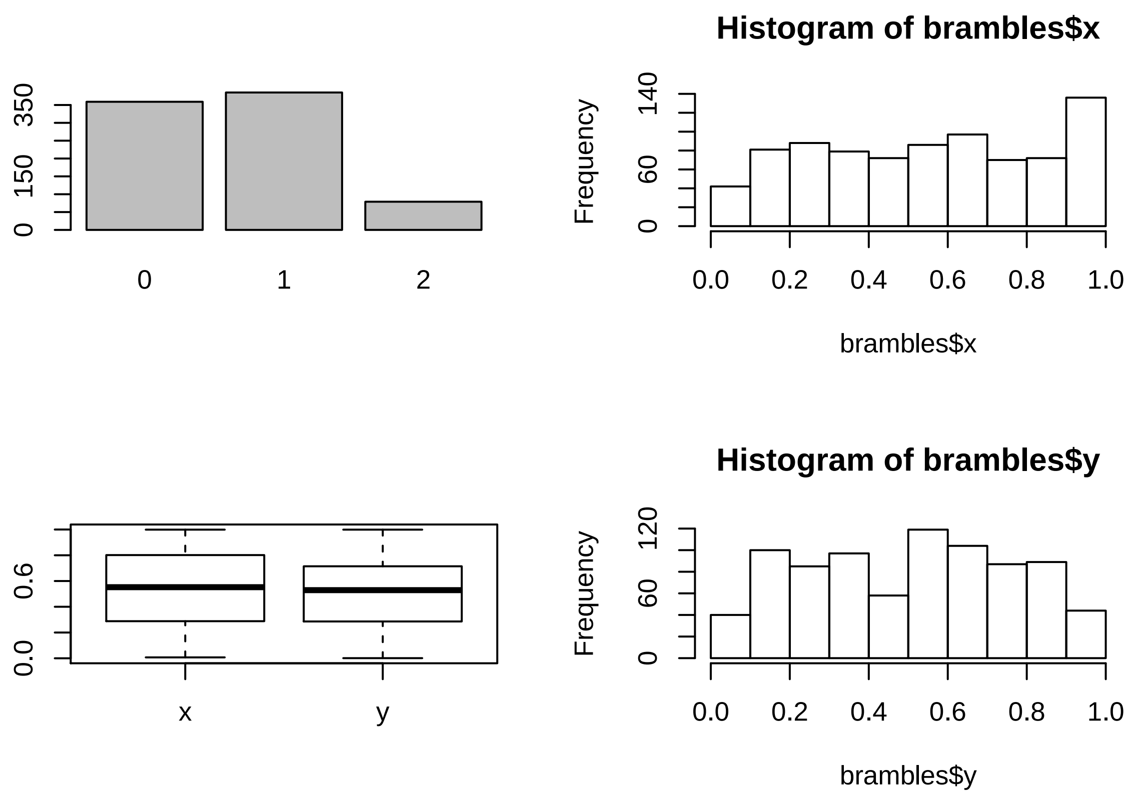
```
x <- seq(from = 1, to = 100) # equal to 1:100
y <- seq(from = 1, to = 1000, by = 10)
plot(y~x, type = "l") # create a line plot
lines(y-200~x, col = "tomato") # add a new line
abline(h = 500, lty = 2) # add a horizontal line with a new linetype
```



Multiple plots in the same window

You can modify the options of the graphics window to display multiple plots at the same time. You do this with the `par(mfrow = c(rows,columns))` command, replacin `rows` and `columns` with suitable integers. To revert your options, close the graphics window or run the command `par(mfrow = c(1,1))`.

```
par(mfrow = c(2,2))
barplot(ages)
hist(brambles$x)
boxplot(brambles[,c("x","y")])
hist(brambles$y)
```



```
par(mfrow = c(1,1))
```

Saving plots

To export your plots from R, use the commands `png()` to save as raster graphics, and `pdf()` to save as vector graphics. Prefer vector graphics, as they generally take less space and retain their resolution.

The `png()` takes as its arguments the width and height of the image in pixels. The default size is quite small, so do adjust.

```
png(filename = "enigmatic_expression.png", width = 700, height = 700)
plot(x = c(2,8), y = c(7,7), xlim = c(0,10), ylim = c(0,10), col = "blue")
lines(c(6,4,3,4,6)~c(1,3,5,7,9), col = "red")
dev.off()
```

```
## pdf
##   2
```

The `pdf()` command's width and height arguments are in inches by default.

```
pdf(file = "vital_organ.pdf", width = 7, height = 7)
plot(x = c(1:9),
     y = c(7,8.5,9,8,7,8,9,8.5,7),
     xlim = c(0,10),
     col = "red",
     type = "l",
     ylim = c(1,10))
lines(x = c(1,5,9), y = c(7,1,7) , col = "red")
dev.off()
```

```
## pdf
##   2
```

Important summary statistics

We have already been introduced to means, medians, and quartiles. We will now take a closer look of **quantiles**, **standard deviations**

Generating data

It is surprising how often one needs to generate data. When producing random data, they are usually sampled from some *distribution*. In this excercise we will cover the uniform and normal distributions.

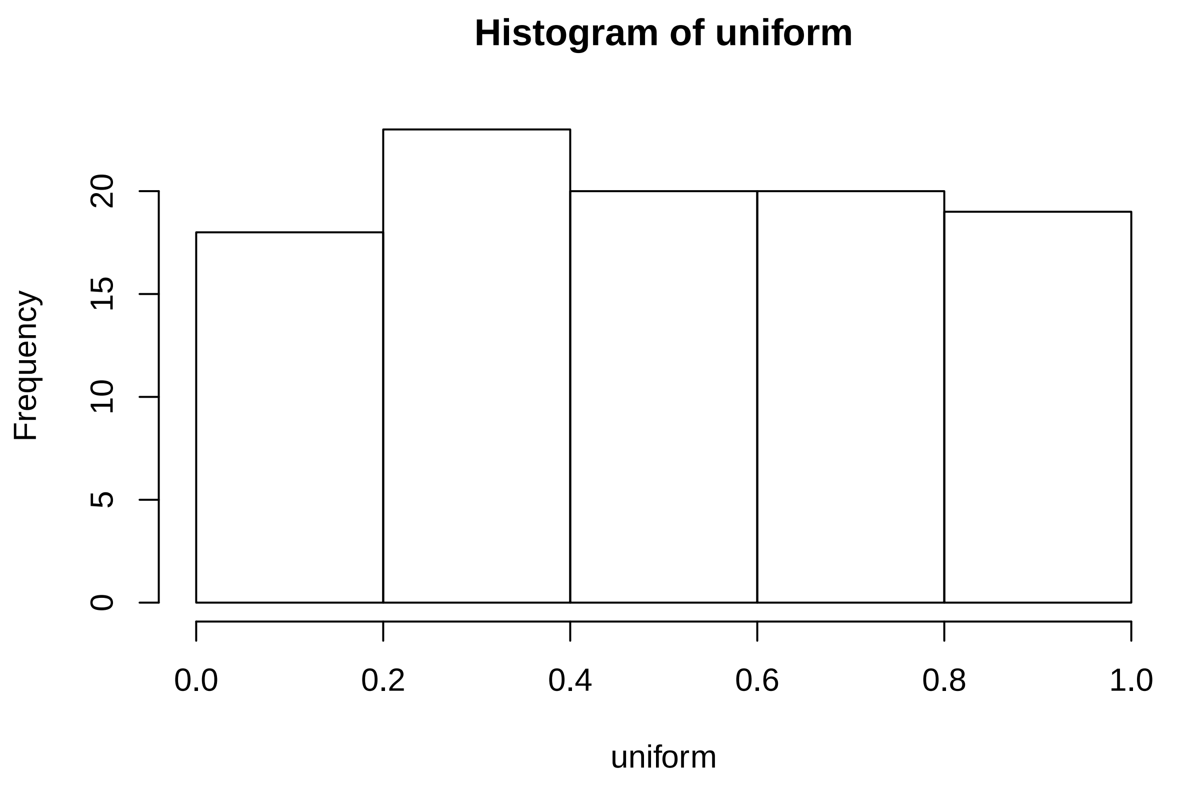
Uniform distribution

The uniform distribution is specified with only two parameters: the **minimum** and the **maximum**. As the name implies, the probability distribution between these two points is uniform; all values have exactly equal probability of being represented in the sample. Random samples from a uniform distribution are produced with the `runif()` command.

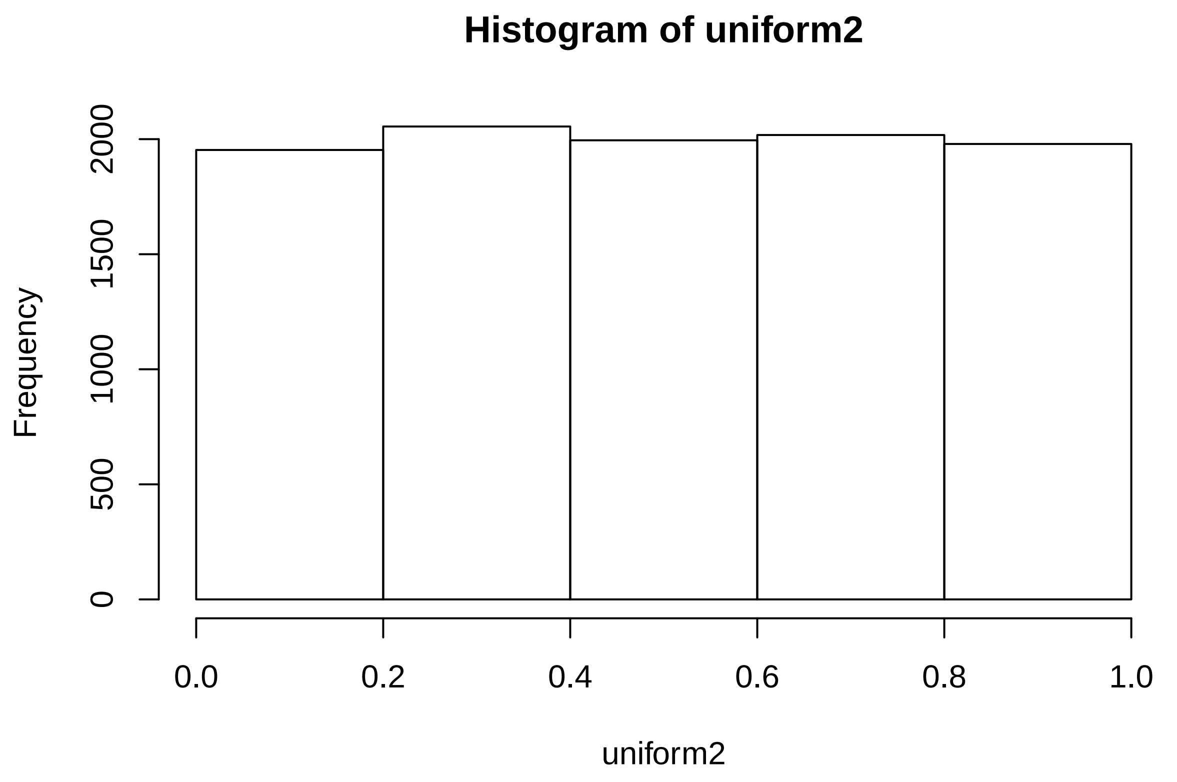
```
## 100 draws from a uniform(0,1) distribution.
uniform <- runif(n = 100, min = 0, max = 1)

## 10000 draws from a uniform(0,1) distribution.
uniform2 <- runif(n = 10000, min = 0, max = 1)

## The sample distribution is not exactly uniform because of sampling error
hist(uniform, breaks = 5)
```



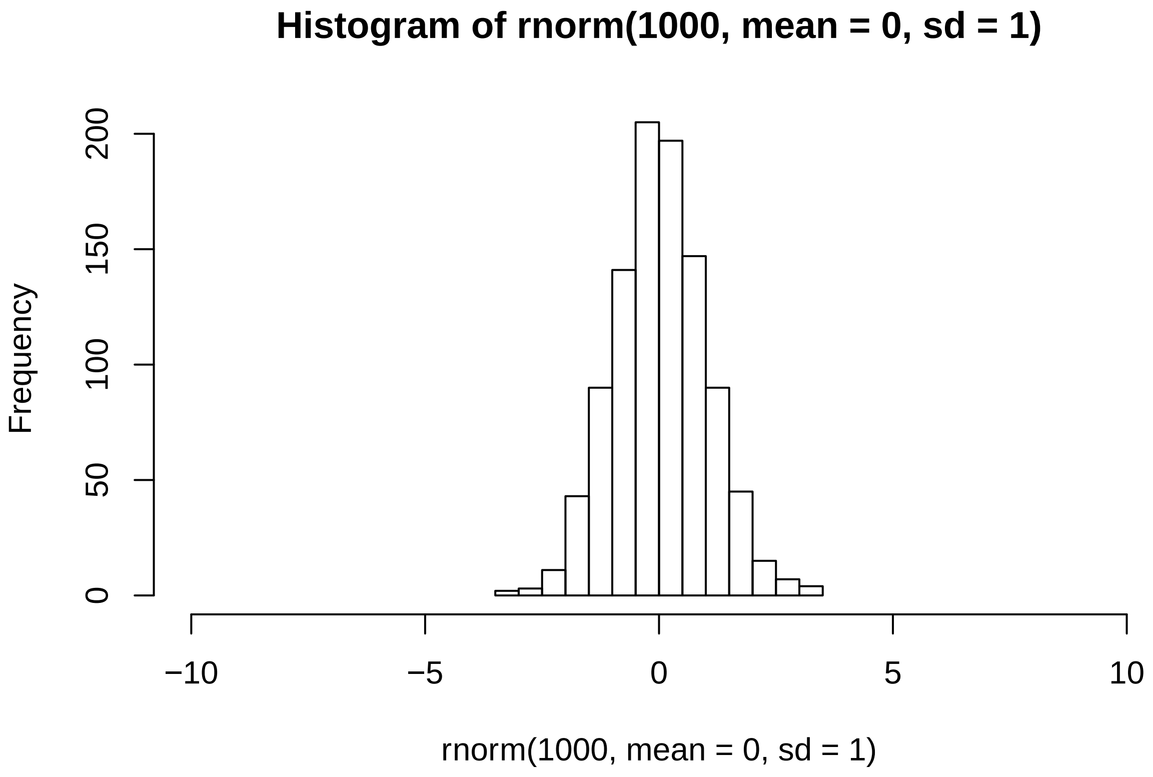
```r
hist(uniform2, breaks = 5) # much closer
```



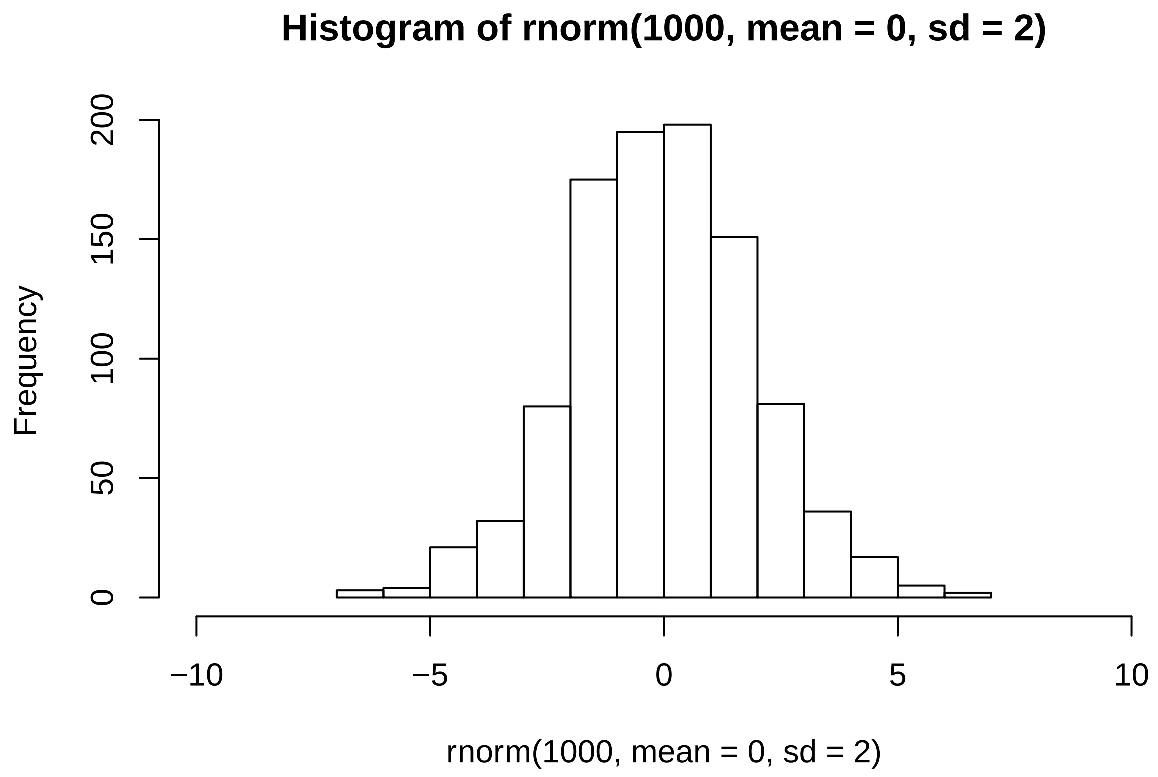
Normal distribution

The normal distribution is one of the most important distributions there is. It's probability distribution is the famous Gauss' bell-shaped curve. The normal distribution is parametrized with a mean and standard deviation parameter. The standard deviation is a *scale parameter*, which defines the variability of sampled values. Random normal deviates are produced with the command `rnorm()`.

```
hist(rnorm(1000, mean = 0, sd = 1), xlim = c(-10,10)) # 1000 normal deviates, sd = 1
```



```
hist(rnorm(1000, mean = 0, sd = 2), xlim = c(-10,10)) # same, but sd 2. The distribution is wider
```



Scaling variables

Sampling

. . . without replacement

. . . with replacement

Bootstrap confidence intervals