Exercises part 1

Course: Data analysis and visualization using R

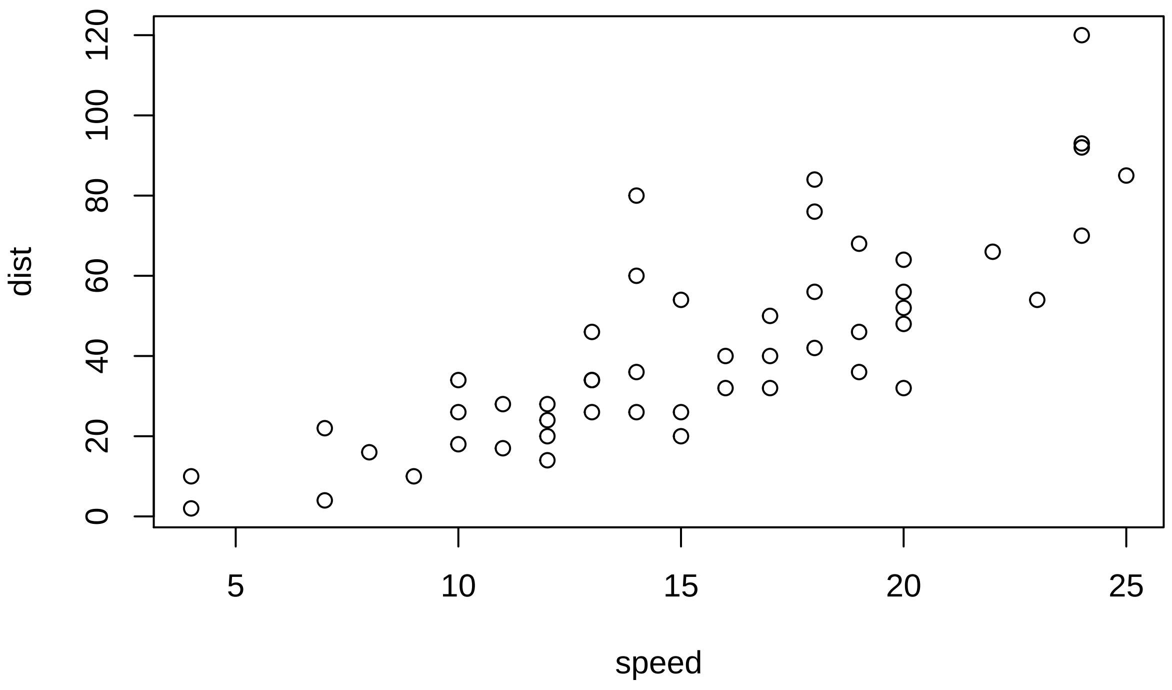
This R Markdown document contains exercises to accompany the course "Data analysis and visualization using R".
This document contains the exercises themselves plus (in most cases) a R code chunk to complete, correct or create. There are some code chucks here bounded with this line: `###### TEST CODE - DO NOT REMOVE OR EDIT #######`. These code chunks are used to give you feedback on the correctness of your implementations. Do not edit or remove these chunks (unless you hate feedback). For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```r
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00  
```

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Before Knitting this document, check if you have the devtools package installed, by typing `library(devtools)` in the console. If this fails, you need to install it by typing `install.packages("devtools")`.

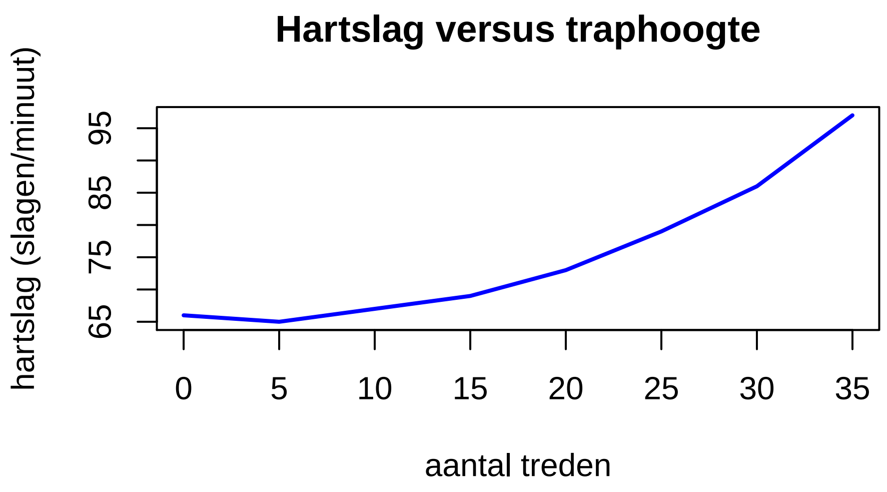
Section 1. Basic plotting

Calculate/carry out the following. With all plots, take care to adhere to the rules regarding titles and other decorations. Tip: the site Quick-R has nice detailed information with examples on the different plot types and their configuration. Especially the section on plotting is helpful for these assignments.

Exercise 1.1.

The vectors below hold data for a staircase walking experiment. A subject of normal weight and height was asked to ascend a (long) stairs wearing a heart-rate monitor. The subjects' heart was registered for different step heights. Create a line (!) plot showing the relationship between heart rate and stair height.

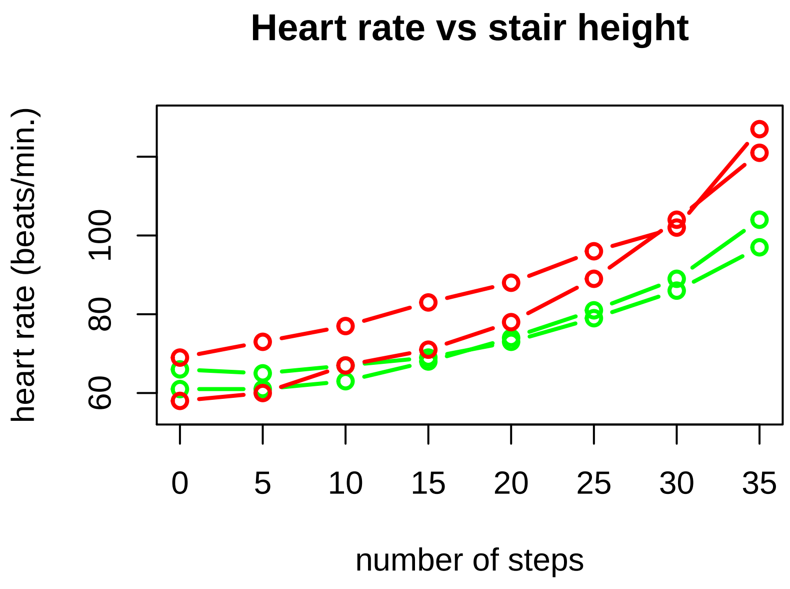
```r
#number of steps on the stairs
stair_height <- c(0, 5, 10, 15, 20, 25, 30, 35)
#heart rate after ascending the stairs
heart_rate <- c(66, 65, 67, 69, 73, 79, 86, 97)
##your code here creating the plot

plot(heart_rate ~ stair_height,
     main = "Hartslag versus traphoogte",
     xlab = "aantal treden",
     ylab = "hartslag (slagen/minuut)",
     type = "l",
     lwd = 2,
     col = "blue")
```



Exercise 1.2.

The experiment from the previous question was extended with three more subjects. One of these subjects was also of normal weight, while two of the subjects were obese. The data are given below. Create a single scatter plot with connector lines between the points showing the data for all four subjects. Give the normal-weighted subjects a green line/marker and the obese subjects a red line/marker. You can add new data series to a plot by using the `points(x, y)` function. Use the `ylim()` function to adjust the Y-axis range.

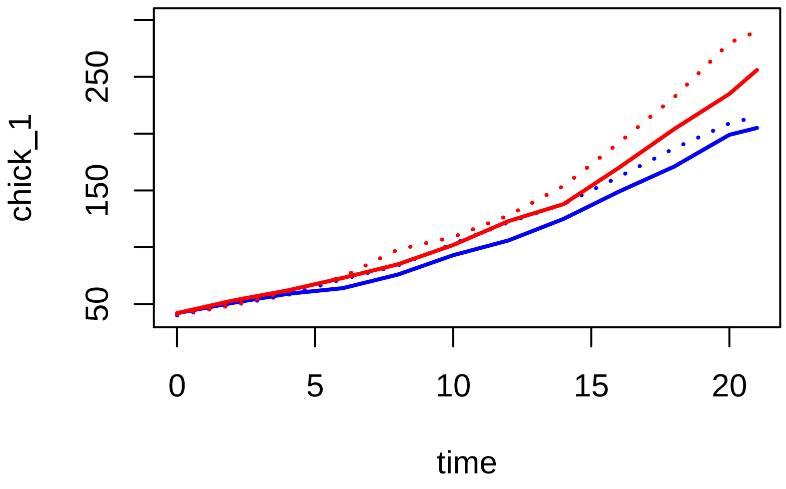
```r
#number of steps on the stairs
stair_height <- c(0, 5, 10, 15, 20, 25, 30, 35)
#heart rates for subjects with normal weight
heart_rate_1 <- c(66, 65, 67, 69, 73, 79, 86, 97)
heart_rate_2 <- c(61, 61, 63, 68, 74, 81, 89, 104)
#heart rates for obese subjects
heart_rate_3 <- c(58, 60, 67, 71, 78, 89, 104, 121)
heart_rate_4 <- c(69, 73, 77, 83, 88, 96, 102, 127)

##your code here creating the plot

plot(x = stair_height,
     y = heart_rate_1,
     main = "Heart rate vs stair height",
     xlab = "number of steps",
     ylab = "heart rate (beats/min.)",
     type = "b",
     lwd = 2,
     col = "green",
     ylim = c(55, 130))
points(x = stair_height,
     y = heart_rate_2,
     col = "green",
     type = "b",
     lwd = 2)
points(x = stair_height,
     y = heart_rate_3,
     col = "red",
     type = "b",
     lwd = 2)
points(x = stair_height,
     y = heart_rate_4,
     col = "red",
     type = "b",
     lwd = 2)
```



Exercise 1.3.

The body weights of chicks were measured at birth and every second day thereafter until day 20. They were also measured on day 21. There were four groups on chicks on different protein diets. Here are the data for the first four chicks. Chick one and two were on diet 1 and chick three and four on diet 2. Create a single line plot showing the data for all four chicks. Give each chick its own color

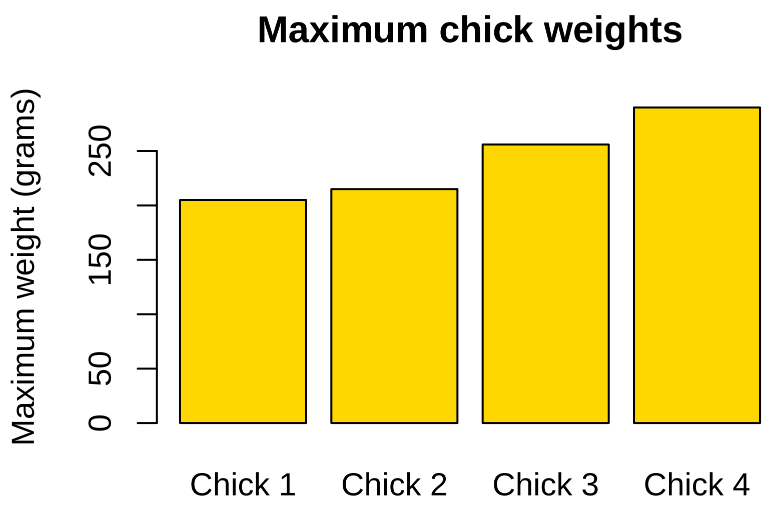
```r
# chick weight data
time <- c(0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 21)
chick_1 <- c(42, 51, 59, 64, 76, 93, 106, 125, 149, 171, 199, 205)
chick_2 <- c(40, 49, 58, 72, 84, 103, 122, 138, 162, 187, 209, 215)
chick_3 <- c(42, 53, 62, 73, 85, 102, 123, 138, 170, 204, 235, 256)
chick_4 <- c(41, 49, 61, 74, 98, 109, 128, 154, 192, 232, 280, 290)

##your code here creating the plot
plot(x = time, y = chick_1,
     type = "l",
     lwd = 2,
     col = "blue",
     ylim = c(40, 300))
points(x = time, y = chick_2,
     type = "l",
     lwd = 2,
     lty = 3,
     col = "blue")
points(x = time, y = chick_3,
     type = "l",
     lwd = 2,
     lty = 1,
     col = "red")
points(x = time, y = chick_4,
     type = "l",
     lwd = 2,
     lty = 3,
     col = "red")
```



Exercise 1.4.

With the data from the previous question, create a barplot of the maximum weights of the chicks.

```r
maxima <- c(max(chick_1), max(chick_2), max(chick_3), max(chick_4))
##"names"" also OK for "names.arg"
barplot(maxima,
  names.arg = c("Chick 1","Chick 2","Chick 3","Chick 4"),
  ylab = "Maximum weight (grams)",
  col = "gold",
  main = "Maximum chick weights")
```



Exercise 1.5.

The R language comes with a wealth of datasets for you to use as practice materials. We will see many of these. One of these datasets is The Time-Series dataset called `discoveries` holding the numbers of "great" inventions and scientific discoveries in each year from 1860 to 1959. Create plot(s) answering these two questions:

(a) What is the frequency distribution of numbers of discoveries per year?

(b) What is the 5-number summary of discoveries per year?

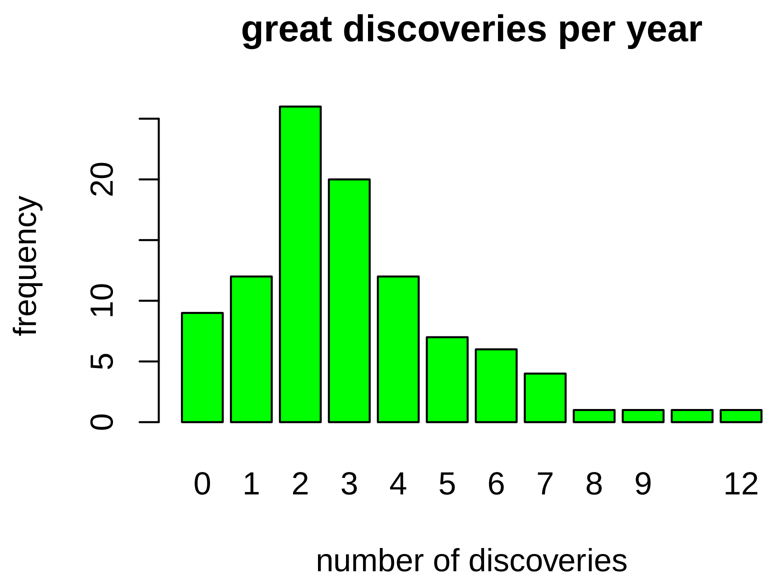
(c) What is the trend over time for the numbers of discoveries per year?

PS actually this is not a simple vector, but a vector with some time=-related attributes called a Time-Series (a `ts` class), but this does not really matter for this assignment.

```r
#load datasets, if not already loaded
library(datasets)
#look at the discoveries dataset
discoveries
```

```
## Time Series:
## Start = 1860
## End = 1959
## Frequency = 1
##   [1]  5  3  0  2  0  3  2  3  6  1  2  1  2  1  3  3  3  5  2  4  4  0  2
##  [24]  3  7 12  3 10  9  2  3  7  7  2  3  3  6  2  4  3  5  2  2  4  0  4
##  [47]  2  5  2  3  3  6  5  8  3  6  6  0  5  2  2  2  6  3  4  4  2  2  4
##  [70]  7  5  3  3  0  2  2  2  1  3  4  2  2  1  1  1  2  1  4  4  3  2  1
##  [93]  4  1  1  1  0  0  2  0
```

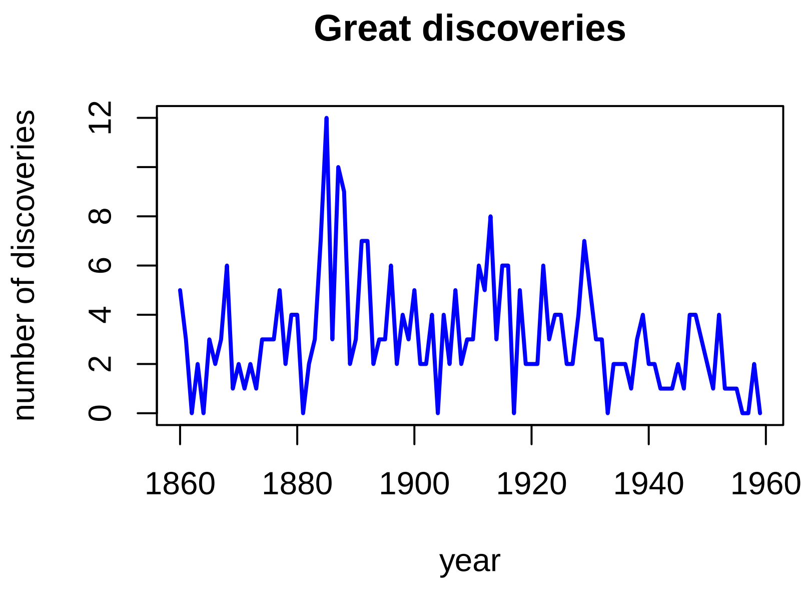
```r
## Q: a
barplot(table(discoveries),
  main = "great discoveries per year",
  xlab = "number of discoveries",
  ylab = "frequency",
  col = "green")
```



```r
##Q b
summary(discoveries)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##     0.0     2.0     3.0     3.1     4.0    12.0
```

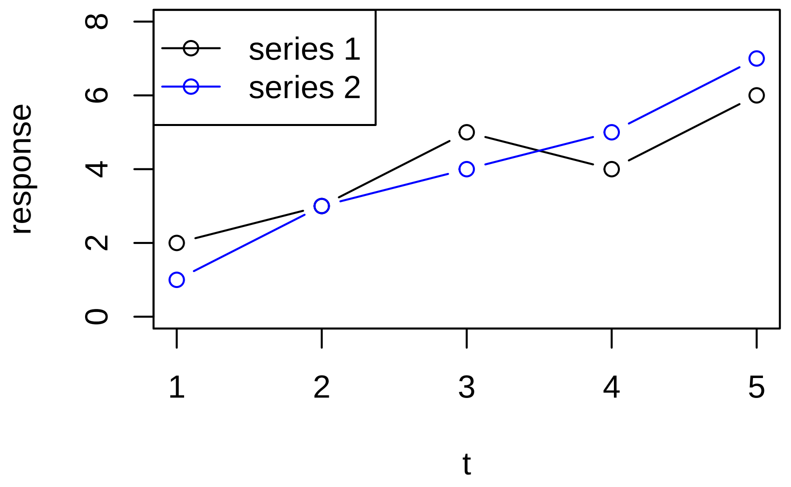
```r
##Q c
plot(discoveries,
     xlab = "year",
     ylab = "number of discoveries",
     main = "Great discoveries",
     col = "blue",
     lwd = 2)
```



Exercise 1.6.

The R datasets package has three related timeseries datasets relating to lung cancer deaths. These are `ldeaths`, `mdeaths` and `fdeaths` for total, male and female deatchs, respectively. Create a line plot showing the montly mortality holding all three of these datasets. Use the `legend()` function to add a legend to the plot, as shown in this example:

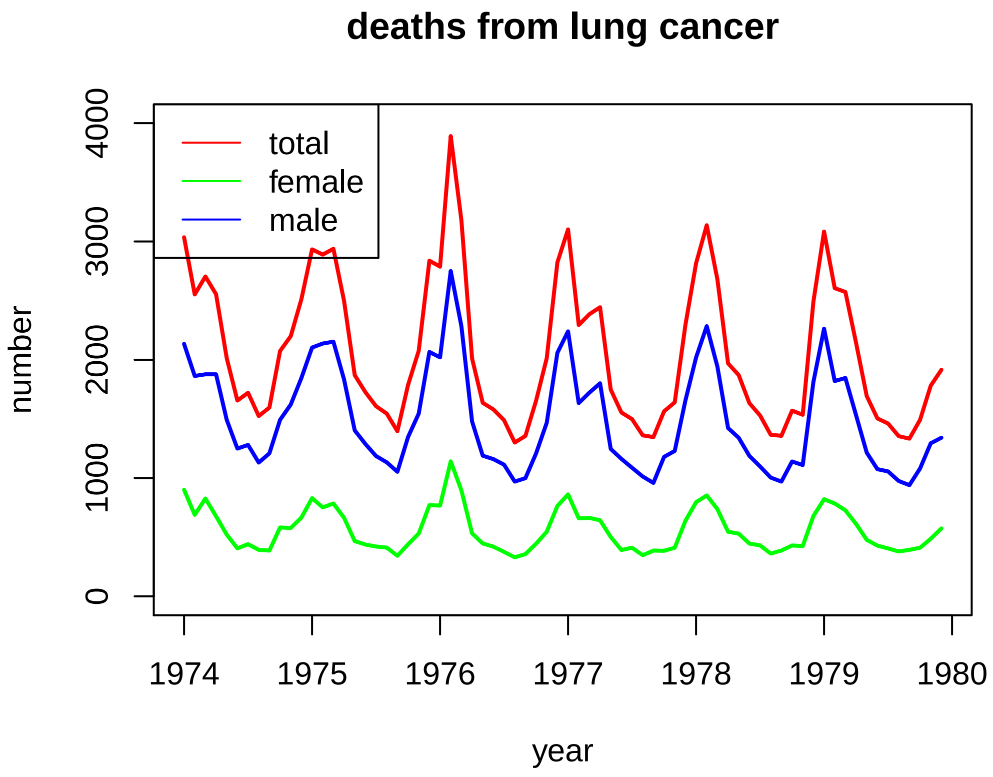
```r
t <- 1:5
y1 <- c(2, 3, 5, 4, 6)
y2 <- c(1, 3, 4, 5, 7)
plot(t, y1, type = "b", ylab = "response", ylim = c(0, 8))
points(t, y2, col = "blue", type = "b")
legend("topleft", legend = c("series 1", "series 2"), col = c("black", "blue"), pch = 1, lty = 1)
```



(a) Create the mentioned line plot. Do you see trends and/or patterns and if so, can you explain these?

(b) Create a combined boxplot of the three time-series. Are there outliers? If so, can you figure out when this occurred?

```r
#load datasets, if not already loaded
library(datasets)

#create plot
total.col <- "red"
m.col <- "blue"
f.col <- "green"
plot(ldeaths,
     main = "deaths from lung cancer",
     xlab = "year",
     ylab = "number",
     col = total.col,
     ylim = c(0, 4000),
     lwd = 2
)
lines(fdeaths, col = f.col, lwd = 2)
lines(mdeaths, col = m.col, lwd = 2)
legend(
  "topleft",
  legend = c("total", "female", "male"),
  col = c(total.col, f.col, m.col),
  lty = 1)
```



Section 2. Complex datatypes & Basic (dataframe) functions

This section serves you some nice datatype challenges. These assignments focus mainly on Dataframes since they are the most important ones.

Exercise 2.1.

Almost all programming languages know the (hash)map / dict data structure storing key and value pairs. R does not have a dict, but you could make an dict-like structure using a list with named elements. Let's have a go at it.

If I wanted to create and use a DNA codon translation table, I could do something like what is shown below. See if you can figure out what is going on there

```r
## define codon table as list
codons <- list(GGA = "Gly", CCU = "Pro", AAA = "Lys", AGU = "Ser")
## the DNA to translate
my.DNA <- "GGACCUAAAAGU"
my.prot <- ""
## iterate the DNA and take only every position
for (i in seq(1, nchar(my.DNA), by=3)) {
    codon <- substr(my.DNA, i, i+2);
    my.prot <- paste(my.prot, codons[[codon]])
}
print(my.prot)
```

```
## [1] " Gly Pro Lys Ser"
```

(a) Make a modified copy of this code chunk in such a way that no spaces are present between the amino acid residues (use help on paste() to figure this out) and that single-letter codes are used instead of three-letter codes.

```r
## define codon table as list
codons <- list(GGA = "G", CCU = "P", AAA = "K", AGU = "S")
## the DNA to translate
my.DNA <- "GGACCUAAAAGU"
my.prot <- ""
## iterate the DNA and take only every position
for (i in seq(1, nchar(my.DNA), by=3)) {
  codon <- substr(my.DNA, i, i+2);
  my.prot <- paste(my.prot, codons[[codon]], sep = "")
}
print(my.prot)
```

```
## [1] "GPKS"
```

(b) Create a list called 'nuc.weights' with named elements containing two vectors - a vector with the Nucleotide single letter codes (A, C, G, T) and a vector with their molecular weights (491.2, 467.2, 507.2, 482.2). Make these list elements accessible through the names 'nucs' and 'weights'. Then iterate `my.DNA` and calculate its molecular weight.

```r
nuc.weights <- list(
    nucs = c("A", "C", "G", "U"),
    weights = c(491.2, 467.2, 507.2, 482.2))
DNA.weight <- 0
for (i in 1:nchar(my.DNA)) {
    current.nuc <- substr(my.DNA, i, i)
    current.nuc.weight <- nuc.weights$weights[nuc.weights$nucs == current.nuc]
    DNA.weight <- DNA.weight + current.nuc.weight
}
DNA.weight
```

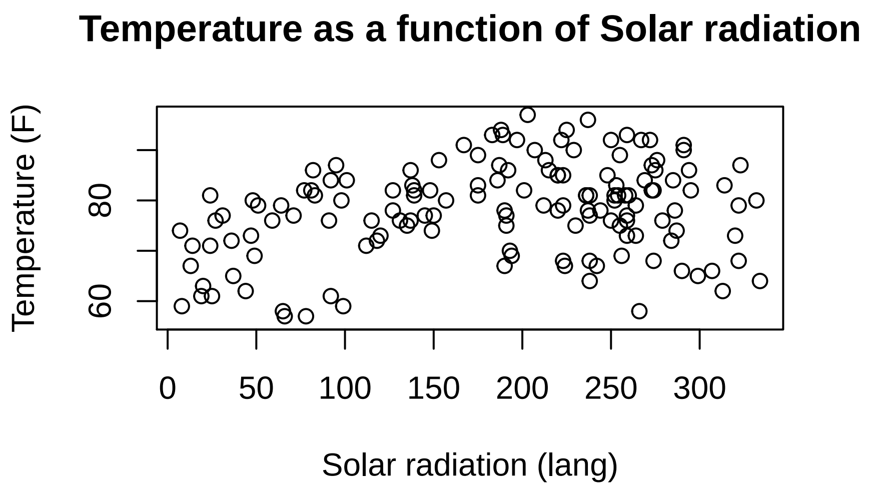
```
## [1] 5876.4
```

Exercise 2.2.

The `airquality` dataset is a dataset included in the `datasets` package. We'll explore this in a few questions.

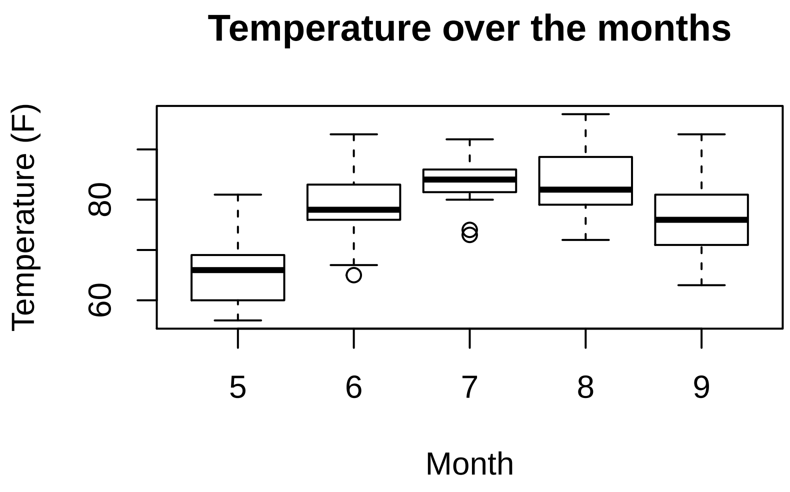
(c) Create a scatterplot of Temperature as a function of Solar radiation. Is there, as you might naively expect, a strong correlation?

```
plot(airquality$Solar.R, airquality$Temp,
     main = "Temperature as a function of Solar radiation",
     xlab = "Solar radiation (lang)",
     ylab = "Temperature (F)")
```



(a) Create a boxplot (-panel) of Temp as a function of Month (use `?boxplot` to find out how this works). What appears to be the warmest month?

```
boxplot(airquality$Temp ~ airquality$Month,
     main = "Temperature over the months",
     xlab = "Month",
     ylab = "Temperature (F)")
```



(b) What date (day/month) has the lowest recorded temperature? Which the highest? Please give temperature values in Celcius, not Fahrenheit! (Yes, this is an extra challenge!)

```
#first create Temp Celcius column:
#(°F  -  32)  x  5/9 = °C
airquality$Temp.C <- (airquality$Temp - 32) * 5/9
#get the required data
airquality[airquality$Temp.C == min(airquality$Temp.C), c("Temp.C", "Month", "Day")]
```

```
##     Temp.C Month Day
## 5 13.33333     5   5
```

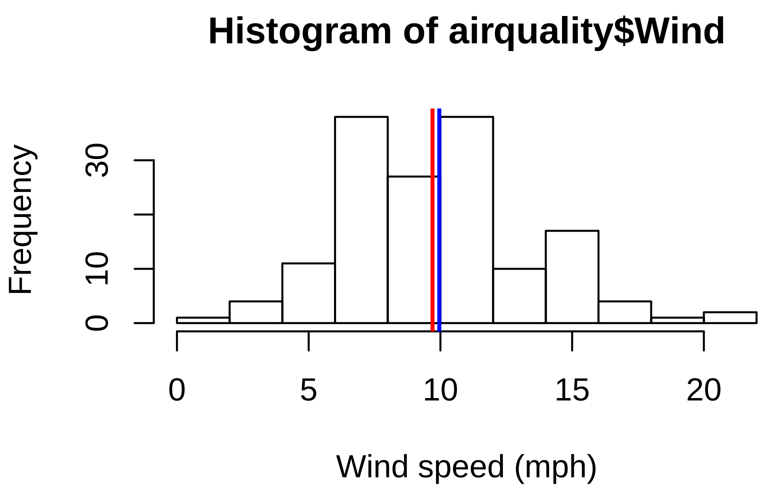
(c) Report the 5 days with highest Ozone observations.

```
airquality[order(airquality$Ozone, decreasing = T)[1:5], ]
```

```
##     Ozone Solar.R Wind Temp Month Day   Temp.C
## 117   168     238  3.4   81     8  25 27.22222
## 62    135     269  4.1   84     7   1 28.88889
## 99    122     255  4.0   89     8   7 31.66667
## 121   118     225  2.3   94     8  29 34.44444
## 30    115     223  5.7   79     5  30 26.11111
```

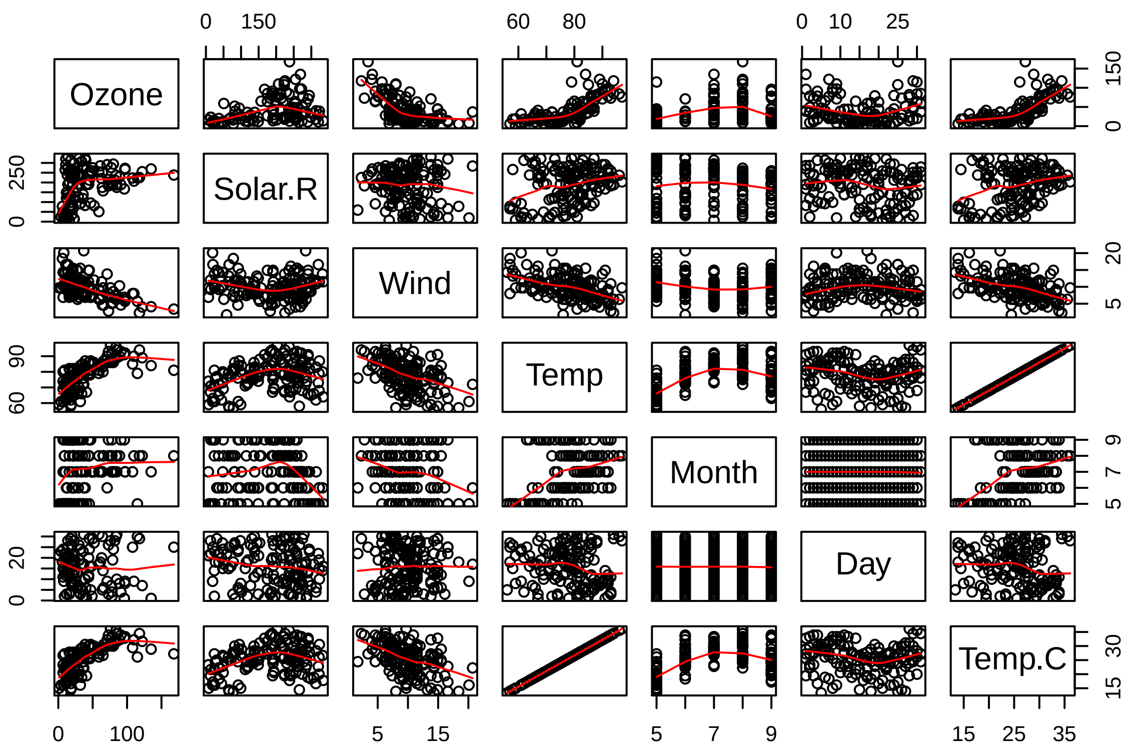
(d) Create a histogram of the wind speeds, and add a fat blue vertical line for the value of the mean and a fat red line for the median (use `abline()` for this).

```
hist(airquality$Wind, xlab = "Wind speed (mph)")
abline(v = mean(airquality$Wind), col = "blue", lwd = 2)
abline(v = median(airquality$Wind), col = "red", lwd = 2)
```



(e) Use the `pairs()` function with argument `panel = panel.smooth` to plot all pairwise correlations between Ozon, Solar radiation, Wind and Temperature. Which pair shows the strongest correlation?

```r
pairs(airquality, panel = panel.smooth)
```



Exercise 2.3.

You will explore a bird observation dataset, downloaded from GOLDEN GATE AUDUBON SOCIETY. This file lists bird observations collected by this bird monitoring group in the San Fransisco Bay Area. You can download and open this file yourself. First open the file in Excel, replace all occurrences of the ";" character to "," and use "Save as.." to save it as ".csv" file (Comma-separated). It is often useful, when working on a project, to set the working directory (type `getwd()` to find out where this is and change it to a more suitable location using `setwd()`)

```
## opens the file. Note the use of the different method arguments! Can you explain these?
bird.obs <- read.table("data/Observations-Data-2014.csv", sep=";", head=T, na.strings = "", quote = "",
```

From here on, it is assumed that you have the dataframe `bird.obs` loaded. This series of exercises deals with cleaning and transforming data, and exploring a cleaned dataset using basic plotting techniques and descriptive statistics.

(a) First, explore the raw data as they are.

- What data on bird observations were recorded (i.e. what kind of variables do you have)?
- What did R do to the original column names (from Observations-Data-2014.xlsx)?
- Are all column names clear to you?

```
## look at the loaded data structure
str(bird.obs)
```

```
## 'data.frame':    2019 obs. of  13 variables:
##  $ Species..  : Factor w/ 325 levels "100","101","102",..: 153 153 153 153 153 270 310 310 310 310 .
##  $ Genus      : Factor w/ 166 levels "Accipiter","Agelaius",..: 8 8 8 8 8 38 38 38 38 38 ...
##  $ Species    : Factor w/ 300 levels "aalge","acuta",..: 11 11 11 11 11 42 235 235 235 235 ...
##  $ Common.name: Factor w/ 329 levels "Acorn Woodpecker",..: 121 121 121 121 121 266 239 239 239 239
##  $ CBRC.Review: Factor w/ 3 levels "FALSE","N","Y": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Date.start : Factor w/ 342 levels "1-Apr-14","1-Aug-14",..: 252 229 14 139 257 117 61 139 307 295
##  $ Date.end   : Factor w/ 229 levels "1-Aug-14","1-Dec-14",..: 74 NA NA NA NA NA NA 131 158 NA ...
##  $ Number     : Factor w/ 70 levels "1","1-2","1-3",..: 1 58 1 1 22 1 1 1 1 1 ...
##  $ Location   : Factor w/ 980 levels " Coyote Creek Trail San Jose",..: 629 639 169 503 28 673 503 5
##  $ County     : Factor w/ 9 levels "Alameda","Contra Costa",..: 7 4 9 9 3 9 9 9 4 4 ...
##  $ Observer.1 : Factor w/ 692 levels "A Sojourner",..: 216 351 544 623 333 623 623 623 323 206 ...
##  $ Other.Obs  : Factor w/ 157 levels "Aaron Maizlish",..: NA NA NA NA NA NA NA NA 155 NA ...
##  $ Notes      : Factor w/ 1262 levels " 3 circling overhead",..: 82 967 NA 1154 830 493 950 949 1096
```

Apparently, all is loaded as a factor; also the Date.start, Date.end (should be dates of course), and Number (should be Integer) and Notes (should be Character) columns. In the original column names tehre are spaces and these are replaced by dots. First column `Species..` is a serial number and the second `Species` is the English species name.

(b) How many bird observations were recorded?

```
nrow(bird.obs)
```

```
## [1] 2019
```

(c) The column holding observation "Number" is actually not a number. What type has R converted it into?

```
class(bird.obs$Number)
```

```
## [1] "factor"
```

(d) Convert the "Number" column into an integer column using `as.integer()`, but assign it to a new column called "Count" (i.e. do not overwrite the original values). Compare the first 50 values or so of these two columns. What happened to the data? Is this OK?

```r
bird.obs$Count <- as.integer(bird.obs$Number)
head(bird.obs[, c(4, 8, 14)], n=50)
```

```
##                       Common.name Number Count
## 1  Greater White-fronted Goose      1     1
## 2  Greater White-fronted Goose      6    58
## 3  Greater White-fronted Goose      1     1
## 4  Greater White-fronted Goose      1     1
## 5  Greater White-fronted Goose      2    22
## 6                   Snow Goose      1     1
## 7                 Ross's Goose      1     1
## 8                 Ross's Goose      1     1
## 9                 Ross's Goose      1     1
## 10                Ross's Goose      1     1
## 11                       Brant    3-6    41
## 12                       Brant      1     1
## 13                       Brant    300    43
## 14                       Brant      1     1
## 15                       Brant      3    36
## 16                       Brant      2    22
## 17                       Brant      9    68
## 18              Cackling Goose      3    36
## 19              Cackling Goose      1     1
## 20              Cackling Goose      1     1
## 21              Cackling Goose      1     1
## 22              Cackling Goose      1     1
## 23              Cackling Goose      3    36
## 24              Trumpeter Swan      6    58
## 25                 Tundra Swan      2    22
## 26                 Tundra Swan      1     1
## 27                 Tundra Swan      2    22
## 28                 Tundra Swan      3    36
## 29                 Tundra Swan      2    22
## 30                 Tundra Swan      1     1
## 31                 Tundra Swan      3    36
## 32                 Tundra Swan      1     1
## 33                 Tundra Swan    145    16
## 34                 Tundra Swan      6    58
## 35                 Tundra Swan     18    21
## 36                 Tundra Swan      3    36
## 37                   Wood Duck      1     1
## 38                     Gadwall      2    22
## 39                     Gadwall      3    36
## 40                     Gadwall      1     1
## 41             Eurasian Wigeon      1     1
## 42             American Wigeon      2    22
## 43             American Wigeon      3    36
## 44             American Wigeon      1     1
## 45             American Wigeon      1     1
## 46             American Wigeon    1-2     2
## 47             American Wigeon    2-5    27
## 48            Blue-winged Teal      3    36
## 49            Blue-winged Teal      1     1
## 50            Blue-winged Teal      1     1
```

This is NOT OK!

(e) The previous question has shown that converting factors to numbers is a bit dangerous. It is often easiest to convert characters to numbers. The best way to do this is by using the `as.is = c(<column indices>)` argument for read.table.

So, which columns should be loaded as real factor data and which as plain character data? Use `read.table()` and the as.is argument to reload the data, and then transform the Number column to integer again.

```
#read with as.is argument
bird.obs <- read.table("data/Observations-Data-2014.csv",
                sep=";",
                head=T,
                na.strings = "",
                quote = "",
                comment.char = "",
                as.is = c(1, 6, 7, 8, 13))
str(bird.obs)
```

```
## 'data.frame':    2019 obs. of  13 variables:
##  $ Species..  : chr  "4" "4" "4" "4" ...
##  $ Genus      : Factor w/ 166 levels "Accipiter","Agelaius",..: 8 8 8 8 8 38 38 38 38 38 ...
##  $ Species    : Factor w/ 300 levels "aalge","acuta",..: 11 11 11 11 11 42 235 235 235 235 ...
##  $ Common.name: Factor w/ 329 levels "Acorn Woodpecker",..: 121 121 121 121 121 266 239 239 239 239
##  $ CBRC.Review: Factor w/ 3 levels "FALSE","N","Y": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Date.start : chr  "3-Jun-14" "28-Jul-14" "1-Sep-14" "2-Sep-14" ...
##  $ Date.end   : chr  "19-Jun-14" NA NA NA ...
##  $ Number     : chr  "1" "6" "1" "1" ...
##  $ Location   : Factor w/ 980 levels " Coyote Creek Trail San Jose",..: 629 639 169 503 28 673 503 5
##  $ County     : Factor w/ 9 levels "Alameda","Contra Costa",..: 7 4 9 9 3 9 9 9 4 4 ...
##  $ Observer.1 : Factor w/ 692 levels "A Sojourner",..: 216 351 544 623 333 623 623 623 323 206 ...
##  $ Other.Obs  : Factor w/ 157 levels "Aaron Maizlish",..: NA NA NA NA NA NA NA NA 155 NA ...
##  $ Notes      : chr  "Adult bird seen on golf course grounds with Canada geese!" "Saw 6 along the sh
```

```
#convert Number column
bird.obs$Count <- as.integer(bird.obs$Number)
```

```
## Warning: NAs introduced by coercion
```

(f) Compare the first 50 values of the Number and Count columns again. Has the conversion succeeded? How many `Number` values could not be transformed into an integer value? Hint: use `is.na()`

```
head(bird.obs[, c(4, 8, 14)], n=50)
```

```
##                  Common.name Number Count
## 1  Greater White-fronted Goose      1     1
## 2  Greater White-fronted Goose      6     6
## 3  Greater White-fronted Goose      1     1
## 4  Greater White-fronted Goose      1     1
## 5  Greater White-fronted Goose      2     2
## 6                   Snow Goose      1     1
## 7                 Ross's Goose      1     1
## 8                 Ross's Goose      1     1
## 9                 Ross's Goose      1     1
## 10                Ross's Goose      1     1
## 11                       Brant    3-6    NA
## 12                       Brant      1     1
## 13                       Brant    300   300
## 14                       Brant      1     1
## 15                       Brant      3     3
## 16                       Brant      2     2
## 17                       Brant      9     9
## 18              Cackling Goose      3     3
## 19              Cackling Goose      1     1
## 20              Cackling Goose      1     1
## 21              Cackling Goose      1     1
## 22              Cackling Goose      1     1
## 23              Cackling Goose      3     3
## 24              Trumpeter Swan      6     6
## 25                 Tundra Swan      2     2
## 26                 Tundra Swan      1     1
## 27                 Tundra Swan      2     2
## 28                 Tundra Swan      3     3
## 29                 Tundra Swan      2     2
## 30                 Tundra Swan      1     1
## 31                 Tundra Swan      3     3
## 32                 Tundra Swan      1     1
## 33                 Tundra Swan    145   145
## 34                 Tundra Swan      6     6
## 35                 Tundra Swan     18    18
## 36                 Tundra Swan      3     3
## 37                   Wood Duck      1     1
## 38                     Gadwall      2     2
## 39                     Gadwall      3     3
## 40                     Gadwall      1     1
## 41             Eurasian Wigeon      1     1
## 42             American Wigeon      2     2
## 43             American Wigeon      3     3
## 44             American Wigeon      1     1
## 45             American Wigeon      1     1
## 46             American Wigeon    1-2    NA
## 47             American Wigeon    2-5    NA
## 48            Blue-winged Teal      3     3
## 49            Blue-winged Teal      1     1
## 50            Blue-winged Teal      1     1
```

```
sum(is.na(bird.obs$Count))
```

```
## [1] 111
```

(g) Explore the sighting counts:

- What is the maximum number of birds in a single sighting? (Use max() and which() or is.na() to solve this)

- What is the mean sighting count
- What is the median of the sighting count

```
#What is the maximum number of birds in a single sighting?
bird.obs[which(bird.obs$Count == max(bird.obs$Count, na.rm = T)), ]
```

```
##     Species..    Genus      Species             Common.name CBRC.Review
## 229       104 Puffinus opisthomelas Black-vented Shearwater           N
##     Date.start Date.end Number   Location    County Observer.1 Other.Obs
## 229  20-Oct-14     <NA>  34000 Moss Beach San Mateo  Ron Thorn      <NA>
##                                  Notes Count
## 229 swirling masses of feeding birds 34000
```

```
##OR
bird.obs[!is.na(bird.obs$Count) & bird.obs$Count == max(bird.obs$Count, na.rm = T), ]
```

```
##     Species..    Genus      Species             Common.name CBRC.Review
## 229       104 Puffinus opisthomelas Black-vented Shearwater           N
##     Date.start Date.end Number   Location    County Observer.1 Other.Obs
## 229  20-Oct-14     <NA>  34000 Moss Beach San Mateo  Ron Thorn      <NA>
##                                  Notes Count
## 229 swirling masses of feeding birds 34000
```

```
#What is the mean sighting count
mean(bird.obs$Count, na.rm = T)
```

```
## [1] 26.79298
```

```
#What is the median of the sighting count
median(bird.obs$Count, na.rm = T)
```

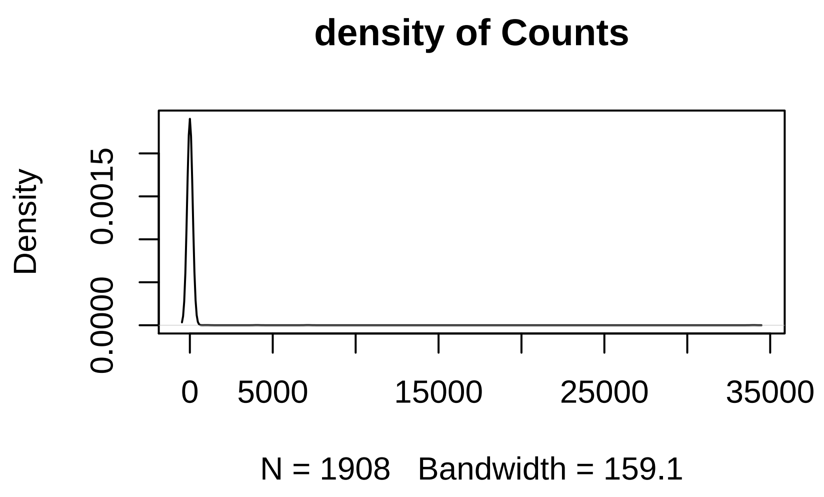
```
## [1] 1
```

Is the Count variable a normal distributed value? You can use `table(...)` of `plot(density(...))` to explore this further.

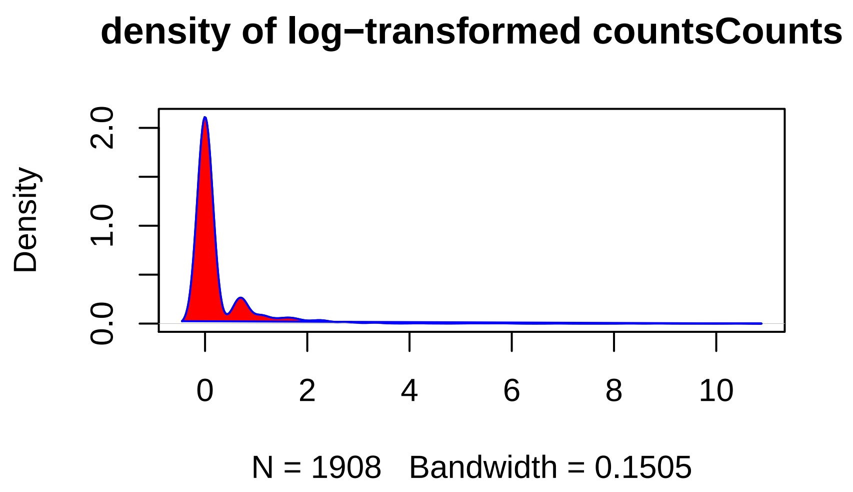
```
table(bird.obs$Count)
```

```
##
##     1     2     3     4     5     6     7     8     9    10    11    12
##  1527   190    55    21    27    19     4     6     8    12     3     1
##    15    16    18    20    25    26    29    30    45    65    94   145
##     8     3     2     1     2     1     1     3     1     2     1     1
##   200   300   400  1000  4050  7120 34000
##     2     2     1     1     1     1     1
```

```
plot(density(bird.obs$Count, na.rm=T),
     main = "density of Counts")
```



```r
##or with a log transformation (and some coloring)
d <- density(log(bird.obs$Count), na.rm=T)
plot(d, main = "density of log-transformed countsCounts")
polygon(d, col = "red", border = "blue")
```



No way this is anywhere near a normal distribution!

(h) Explore the species constitution:

- How many different species were recorded?
- How many genera do they constitute?
- What species from the genus "Puffinus" have been observed?

Hint: use the function `unique()` here.

```r
#How many different species were recorded?
length(unique(bird.obs$Common.name))
```

```
## [1] 330
```

```r
#How many genera do they constitute?
length(unique(bird.obs$Genus))
```

```
## [1] 166
```

```r
#What species from the genus "Puffinus" have been observed?
#the actual sightings
bird.obs[bird.obs$Genus == "Puffinus", c(2, 3, 4, 6, 14)]
```

```
##        Genus      Species            Common.name Date.start Count
## 209 Puffinus    creatopus  Pink-footed Shearwater  18-Jan-14     2
## 210 Puffinus    creatopus  Pink-footed Shearwater  20-Feb-14     3
## 211 Puffinus    carneipes Flesh-footed Shearwater  20-Jul-14     2
## 212 Puffinus    carneipes Flesh-footed Shearwater   6-Sep-14     1
## 213 Puffinus    carneipes Flesh-footed Shearwater  12-Oct-14     3
## 214 Puffinus    carneipes Flesh-footed Shearwater  13-Oct-14     1
## 215 Puffinus      griseus        Sooty Shearwater   3-Dec-15     1
## 216 Puffinus tenuirostris Short-tailed Shearwater  20-Feb-14     2
## 217 Puffinus tenuirostris Short-tailed Shearwater   2-Mar-14     1
## 218 Puffinus     puffinus         Manx Shearwater  26-Sep-14     1
## 219 Puffinus     puffinus         Manx Shearwater  12-Oct-14     1
## 220 Puffinus opisthomelas Black-vented Shearwater   3-Feb-14     2
## 221 Puffinus opisthomelas Black-vented Shearwater  20-Feb-14     9
## 222 Puffinus opisthomelas Black-vented Shearwater  30-May-14     1
## 223 Puffinus opisthomelas Black-vented Shearwater  20-Jul-14     1
## 224 Puffinus opisthomelas Black-vented Shearwater  20-Jul-14     6
## 225 Puffinus opisthomelas Black-vented Shearwater   1-Aug-14     1
## 226 Puffinus opisthomelas Black-vented Shearwater  13-Oct-14   200
## 227 Puffinus opisthomelas Black-vented Shearwater  17-Oct-14   300
## 228 Puffinus opisthomelas Black-vented Shearwater  18-Oct-14  4050
## 229 Puffinus opisthomelas Black-vented Shearwater  20-Oct-14 34000
```

```r
#the species
unique(bird.obs[bird.obs$Genus == "Puffinus", "Common.name"])
```

```
## [1] Pink-footed Shearwater  Flesh-footed Shearwater Sooty Shearwater
## [4] Short-tailed Shearwater Manx Shearwater         Black-vented Shearwater
## 329 Levels: Acorn Woodpecker Allen's Hummingbird ... Yellow-throated Vireo
```

(i) This is a challenge exercise for those who like to grind their brains! Think of a strategy to "rescue" the NAs that appear after transforming "Number" to "Count". Hint: use `gsub()` or`grep()`

```r
#these are the values that need to be rescued:
table(bird.obs[is.na(bird.obs$Count), "Number"])
```

```
## 
##     1-2     1-3     1-4     1-5     1-6     1-7     1-8      1+   10-36 
##      32      11       4       4       1       1       1       1       1 
##    100s     15+   16-24    2-10    2-17     2-3     2-4     2-5     2-6 
##       1       1       1       1       1       5       4       2       1 
##     2-7      2+    3-10    3-11    3-12     3-4     3-6    4-10    4-12 
##        1        1        1        1        1        1        5        1        1
##      4-5      4-6     5-11     5-14     5-20     5-25      6-8     7-12      7-8
##        1        1        1        1        1        1        1        1        1
##     8-11       8+  several
##        1        1        1
```

```
#I suggest you take the lowest of the range-like values: 1-3 becomes 1; 2-3 becomes 2; 100s becomes 100
#then do something like
tmp <- bird.obs$Number[1:50]
tmp
```

```
##  [1] "1"   "6"   "1"   "1"   "2"   "1"   "1"   "1"   "1"   "1"   "3-6"
## [12] "1"   "300" "1"   "3"   "2"   "9"   "3"   "1"   "1"   "1"   "1"
## [23] "3"   "6"   "2"   "1"   "2"   "3"   "2"   "1"   "3"   "1"   "145"
## [34] "6"   "18"  "3"   "1"   "2"   "3"   "1"   "1"   "2"   "3"   "1"
## [45] "1"   "1-2" "2-5" "3"   "1"   "1"
```

```
gsub("(\\d+)-(\\d+)", "\\1", tmp)
```

```
##  [1] "1"   "6"   "1"   "1"   "2"   "1"   "1"   "1"   "1"   "1"   "3"
## [12] "1"   "300" "1"   "3"   "2"   "9"   "3"   "1"   "1"   "1"   "1"
## [23] "3"   "6"   "2"   "1"   "2"   "3"   "2"   "1"   "3"   "1"   "145"
## [34] "6"   "18"  "3"   "1"   "2"   "3"   "1"   "1"   "2"   "3"   "1"
## [45] "1"   "1"   "2"   "3"   "1"   "1"
```

WE WILL REVISIT THIS DATASET LATER ON, WHEN WORKING WITH THE apply FUNCTIONS

Section 3. Flow control & Functions in R

This section serves you some exercises that will help you improve your function-writing skills.

Illegal reproductions

As an exercise, you will re-invent the wheel here for some statistical functions.

Exercise 3.1.

Create a function, `my.mean()`, that duplicates the R function mean(), i.e. calculates and returns the average of a vector of numbers.

```
my.mean <- function(x) {
    sum(x, na.rm = T) / length(x)
}
```

```
## [1] "You rock!"
```

Exercise 3.2.

Create a function, `my.sd()`, that duplicates the R function sd(), i.e. calculates and returns the standard deviation of a vector of numbers.

```
my.sd <- function(x) {
    sqrt(sum((x - mean(x))^2)/(length(x)-1))
}
```

```
## [1] "You rock!"
```

Exercise 3.3.

Create a function, `my.median()`, that duplicates the R function median(), i.e. calculates and returns the median of a vector of numbers. This is actually a bit harder than you might expect.

```
my.median <- function(x) {
    sorted <- sort(x)
    if(length(x) %% 2 == 1) {
        #uneven length
        x.median <- sorted[ceiling(length(x)/2)]
    } else {
        x.median <- (sorted[length(x)/2] + sorted[(length(x)/2)+1]) / 2
    }
    return(x.median)
}
```

```
## [1] "You rock!"
```

Exercise 3.4.

Create a function, `GC.perc()`, that calculates and returns the GC percentage of a DNA or RNA sequence. Accept as input a sequence and a flag `-strict-` indicating whether other characters are accepted than core DNA (GATUC). If `strict = FALSE`, the percentage of other characters should be reported using a `warn()` call. If `strict = TRUE`, the function should terminate with an error message. Use `stop()` for this. `strict` should default to TRUE. NOTE, usage of `strict` can complicate things, so start with the core functionality!

```
GC.perc <- function(seq, strict = TRUE) {
    if (is.na(seq)) {
        return(NA)
    }
    if (length(seq) == 0) {
        return(0)
    }
    seq.split <- strsplit(seq, "")[[1]]
    gc.count <- 0
    anom.count <- 0
    for (n in seq.split) {
        if (length(grep("[GATUCgatuc]", n)) > 0) {
            if (n == "G" || n == "C") {
                gc.count <- gc.count + 1
            }
        } else {
```

```r
            if (strict) {
                stop(paste("Illegal character", n))
            } else {
                anom.count <- anom.count + 1
            }
        }
    }
    ##return perc
    ##print(gc.count)
    if (anom.count > 0) {
        anom.perc <- anom.count / nchar(seq) * 100
        warning(paste("Non-DNA characters have percentage of", anom.perc))
    }
    return(gc.count / nchar(seq) * 100)
}
```

END OF PART ONE