Log 708 - Chapter 3 Solutions

Halvard Arntzen

These are suggested solutions to chapter 3 exercises. In many cases, R offers different ways of doing things, so this is not a list of "definitive" answers. And of course, even though a major part of these exercises is "watching videos", we do not watch the videos for you :-)

3.1

This is something you just have to do on your own!

3.2

c.

```
# assign your year of birth to a variable "year"
year <- 1968

# make a vector "date" = (year, month, day) with your date of birth.
# Recall, there is a super-central R function that creates vectors.
date <- c(1968, 9, 21)

# make a variable "a" that contains the value (2 + 3)*(10 - 3)^2.
a <- (2+3)*(10-3)^2

#assign the value 10 to variable "b" and let z be the sum of a and b.
b <- 10
z <- a + b

#assign the value 1 to "a", 2 to "b", 3 to "c".
#you can just overwrite the existing content in the variables.

a <- 1
b <- 2
c <- 3

#test (using logical operator) whether a is equal to b.
a == b
```

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

```
#test whether a+b = c.
a+b == c
```

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

```
#assign the numbers 12,13,14 and 15 to "f", i.e. make a vector.
f <- c(12,13,14,15)

#make a vector y that contains the product by b of all elements in f.
y <- b*f


#Let u, v be the vectors below:
u <- c(55, 29, 51, 35, 33, 42)
v <- c(2, 4, 6, 8, 10, 12)

#assign the 4th element of u to a variable x
x <- u[4]

#change the first element of u to the value 25
u[1] <- 25

#make a logical vector z with values TRUE/FALSE depending on whether u < 35
z <- (u < 35)

#make a logical vector w with values TRUE/FALSE depending on whether u < 35 and v > 5
w <- (u < 35) & (v > 5)
w
```

```
## [1] FALSE FALSE FALSE FALSE  TRUE FALSE
```

```
#use R functions to find the length and type of u and w
length(u)
```

```
## [1] 6
```

```
str(u)
```

```
##  num [1:6] 25 29 51 35 33 42
```

```
length(w)
```

```
## [1] 6
```

```
str(w)
```

```
##  logi [1:6] FALSE FALSE FALSE FALSE TRUE FALSE
```

```
#check that u+v, u*v, u/v and v^4 gives the proper result in vectorized form

u+v
```

```
## [1] 27 33 57 43 43 54
```

```
u*v
```

```
## [1]  50 116 306 280 330 504
```

```
u/v
```

```
## [1] 12.500  7.250  8.500  4.375  3.300  3.500
```

```
v^4
```

```
## [1]    16   256  1296  4096 10000 20736
```

```
#use ls() to list the global environment. Check that the result is the same as you see
#with the cursor in the script window, hit CTRL+SHIFT+ENTER. This should run ALL comma

ls()
```

```
##  [1] "a"           "A"           "AQ"          "AQorig"      "b"           "B"           "B
##  [8] "c"           "date"        "DF"          "df2"         "df3"         "f"           "f
## [15] "M"           "mtemp"       "my_x"        "pop"         "prob"        "S"           "s
## [22] "solradmean" "t"           "toss"        "tossfac"     "u"           "U"           "v
## [29] "V"           "w"           "x"           "y"           "year"        "z"
```

```
#Save your script file with a sensible name in a sensible place, close it.
#Open it again to verify that your work is all there.
#congratulate yourself with the excellent work, and take a short
#break:-)
```

That shows solutions to most of the questions.

3.3

```
#make a vector z of 5-10 arbitary numbers.

z <- c(5, 67, 33, 43, 41, 55)   # or try z <- sample(1:100, 10)

#Find the sum of elements in z
sum(z)
```

```
## [1] 244
```

```
#Find the average value of z
mean(z)
```

```
## [1] 40.66667
```

```
#Find the median value of z
median(z)
```

```
## [1] 42
```

```
#find the mean value of z^2
mean(z^2)
```

```
## [1] 2026.333
```

```
#In the script editor, write "M <-  med". What happens? Hit TAB. What happens?
#You should get a list of options starting with "med". Select the one you need
#using up/down arrows and enter.

#At the console, write "fact", and use this to find "factorial(10)" which is 10*9*8*..
factorial(10)
```

```
## [1] 3628800
```

```
factorial(100)
```

```
## [1] 9.332622e+157
```

100! is a number starting 9 332 6 .... which has 157 digits.

```
#Suppose X is a standard normal variable. Find the probability that X < 2.5
pnorm(2.5)
```

```
## [1] 0.9937903
```

```
#Find the probability that X > 2.5 (there are two ways, either law of complement or us
1 - pnorm(2.5)
```

```
## [1] 0.006209665
```

```
pnorm(2.5, lower.tail = FALSE)
```

```
## [1] 0.006209665
```

```
#Suppose W is a normal variable with mean 4000, standard deviation 300. Find the proba

pnorm(4500, mean = 4000, sd = 300)
```

```
## [1] 0.9522096
```

```
pnorm(5000, mean = 4000, sd = 300, lower.tail = F)
```

```
## [1] 0.0004290603
```

```
#Find all probabilities that W < x, for x = 4100, 4200, 4300, 4400, 4500 utilizing vec
```

```r
x <- c(4100, 4200, 4300, 4500)
pnorm(x, mean = 4000, sd = 300)
```

```
## [1] 0.6305587 0.7475075 0.8413447 0.9522096
```

```r
#write R code to generate the following sequences

# 1,3,5,7,9,11
seq(from=1, to=11, by=2)
```

```
## [1]  1  3  5  7  9 11
```

```r
# 5,6,7,8,9,10
5:10
```

```
## [1]  5  6  7  8  9 10
```

```r
# 1,3,1,3,1,3,1,3,1,3
rep(c(1,3), 5)   #or rep(c(1,3), times = 5)
```

```
##  [1] 1 3 1 3 1 3 1 3 1 3
```

```r
# 8,7,6,5,4,3,2,1
8:1
```

```
## [1] 8 7 6 5 4 3 2 1
```

```r
#suppose y is the vector defined by
y <- c(5, 6, 12, 53, 15, 95, 12, 51, 25, 34, 59, 15, 40, 34)

#find the length of y
length(y)
```

```
## [1] 14
```

```r
#extract a vector with the last 6 elements of y
y[9:14]
```

```
## [1] 25 34 59 15 40 34
```

3.4

a.

```r
DF <- data.frame(name = c("Norway", "Sweden", "Denmark", "Finland",
                          "Germany", "Holland", "Switzerland", "Poland"),
                 population = c(5.5, 10.7, 5.9, 5.5, 84.4, 17.6, 8.8, 40),
                 GDP = c(593, 593, 400, 300, 4456, 991, 885, 1801))
```

```
#view the dataframe in the console - and in a spreadsheet-like view.
DF
```

```
##          name population  GDP
## 1      Norway        5.5  593
## 2      Sweden       10.7  593
## 3     Denmark        5.9  400
## 4     Finland        5.5  300
## 5     Germany       84.4 4456
## 6     Holland       17.6  991
## 7 Switzerland        8.8  885
## 8      Poland       40.0 1801
```

Use `View(DF)` for the spreadsheet view. (Note capital "V" here.)

```
#use R functions to find number of rows, number of columns.
c(nrow(DF), ncol(DF))
```

```
## [1] 8 3
```

```
#get a vector with the names of variables in DF.
names(DF)
```

```
## [1] "name"       "population" "GDP"
```

```
#play with head and tail to show first and last 3 lines of DF.
head(DF, 3)
```

```
##      name population GDP
## 1  Norway        5.5 593
## 2  Sweden       10.7 593
## 3 Denmark        5.9 400
```

```
tail(DF, 3)
```

```
##          name population  GDP
## 6     Holland       17.6  991
## 7 Switzerland        8.8  885
## 8      Poland       40.0 1801
```

```
#rename column "GDP" to "gdp"
names(DF)[3] <- "gdp"

#Find the median GDP and the mean population in DF.
median(DF$gdp)
```

```
## [1] 739
```

```
mean(DF$population)
```

```
## [1] 22.3
```

Also `with(DF, median(gdp))` works here.

```
#Extract the column "population" into a separate vector "pop" (outside of DF)
pop <- DF$population

#Extract subset A of DF with countries of population greater than 8 million
A <- subset(DF, population > 8)
A
```

```
##          name population  gdp
## 2      Sweden       10.7  593
## 5     Germany       84.4 4456
## 6     Holland       17.6  991
## 7 Switzerland        8.8  885
## 8      Poland       40.0 1801
```

```
#Extract subset B of DF where population is greater than 8 and gdp < 1000.
B <- subset(DF, population > 8 & gdp < 1000)
B
```

```
##          name population gdp
## 2      Sweden       10.7 593
## 6     Holland       17.6 991
## 7 Switzerland        8.8 885
```

```
#How can you also extract subset B from A?
B2 <- subset(A, gdp < 1000)
B2
```

```
##          name population gdp
## 2      Sweden       10.7 593
## 6     Holland       17.6 991
## 7 Switzerland        8.8 885
```

```
#extract row 5 from DF
DF[5, ]
```

```
##      name population  gdp
## 5 Germany       84.4 4456
```

```
#extract a dataframe with rows 2,3,4 from DF
DF[2:4, ]
```

```
##      name population gdp
## 2  Sweden       10.7 593
## 3 Denmark        5.9 400
## 4 Finland        5.5 300
```

```r
#extract a dataframe with columns 2, 3 from DF
DF[ ,2:3]
```

```
##   population  gdp
## 1        5.5  593
## 2       10.7  593
## 3        5.9  400
## 4        5.5  300
## 5       84.4 4456
## 6       17.6  991
## 7        8.8  885
## 8       40.0 1801
```

```r
#extract a dataframe with columns 1 and 3 from DF
DF[, c(1, 3)]
```

```
##          name  gdp
## 1      Norway  593
## 2      Sweden  593
## 3     Denmark  400
## 4     Finland  300
## 5     Germany 4456
## 6     Holland  991
## 7 Switzerland  885
## 8      Poland 1801
```

```r
#extract a dataframe with rows 1-4 and columns 1 and 3 from DF
DF[1:4, c(1,3)]
```

```
##      name gdp
## 1  Norway 593
## 2  Sweden 593
## 3 Denmark 400
## 4 Finland 300
```

```r
#run this code, but try first to guess what it does:
subset(DF, startsWith(name, "S"))
```

```
##          name population gdp
## 2      Sweden       10.7 593
## 7 Switzerland        8.8 885
```

```r
#compute a new column gdppc, containg GDP per capita for the countries.
DF$gdppc <- DF$gdp / DF$population


#given that population is in millions, and GDP is in billions USD, what is correct uni
```

The unit for gdppc will be 1000USD, so for example Denmark has about 68 000 USD in GDP per capita.

```
#use "summary" function to summarize the data in DF
summary(DF)
```

```
##      name              population         gdp              gdppc
##  Length:8           Min.   : 5.50   Min.   : 300.0   Min.   : 45.02
##  Class :character   1st Qu.: 5.80   1st Qu.: 544.8   1st Qu.: 54.11
##  Mode  :character   Median : 9.75   Median : 739.0   Median : 55.86
##                     Mean   :22.30   Mean   :1252.4   Mean   : 67.53
##                     3rd Qu.:23.20   3rd Qu.:1193.5   3rd Qu.: 75.99
##                     Max.   :84.40   Max.   :4456.0   Max.   :107.82
```

b.

```
#sort by gdp
S <- order(DF$gdp)
S
```

```
## [1] 4 3 1 2 7 6 8 5
```

```
DF[S, ]
```

```
##          name population  gdp     gdppc
## 4     Finland        5.5  300  54.54545
## 3     Denmark        5.9  400  67.79661
## 1      Norway        5.5  593 107.81818
## 2      Sweden       10.7  593  55.42056
## 7 Switzerland        8.8  885 100.56818
## 6     Holland       17.6  991  56.30682
## 8      Poland       40.0 1801  45.02500
## 5     Germany       84.4 4456  52.79621
```

```
#sort by gdp decreasing
U <- order(DF$gdp, decreasing = TRUE)
U
```

```
## [1] 5 8 6 7 1 2 3 4
```

```
DF[U, ]
```

```
##          name population  gdp     gdppc
## 5     Germany       84.4 4456  52.79621
## 8      Poland       40.0 1801  45.02500
## 6     Holland       17.6  991  56.30682
## 7 Switzerland        8.8  885 100.56818
## 1      Norway        5.5  593 107.81818
## 2      Sweden       10.7  593  55.42056
```

```
## 3     Denmark         5.9  400  67.79661
## 4     Finland         5.5  300  54.54545
```

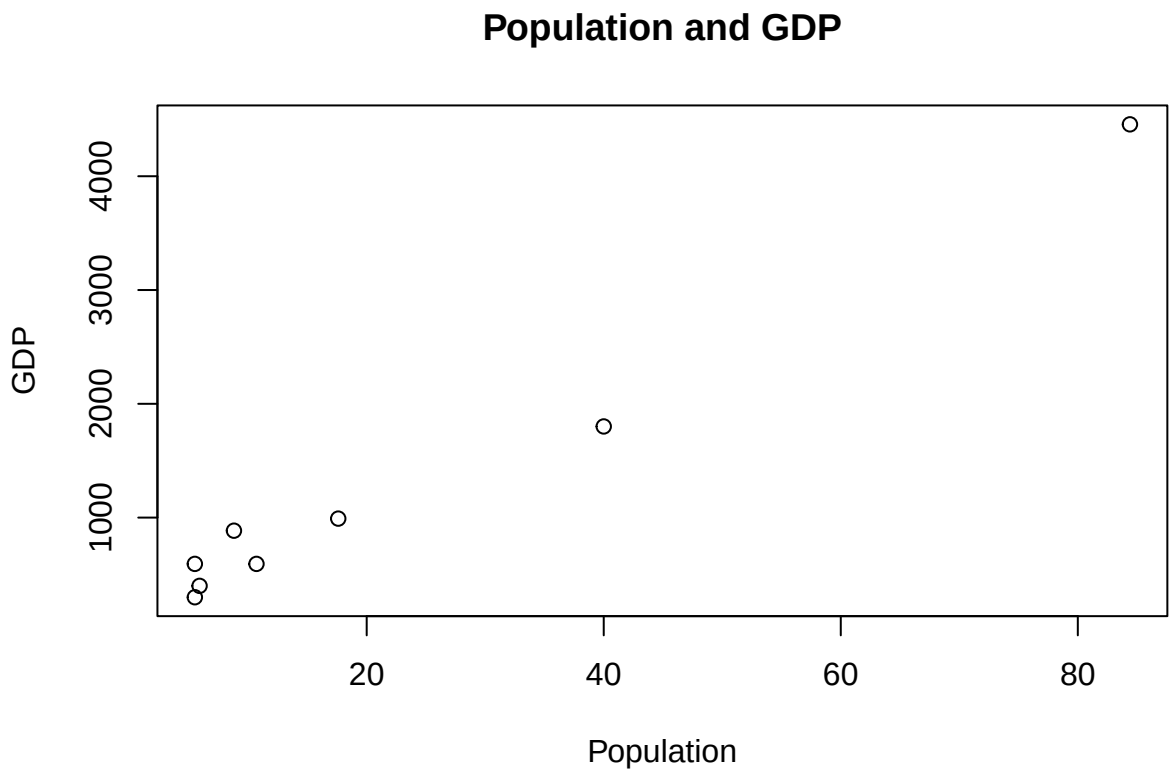
```
#sort by name:
V <- order(DF$name)
DF[V, ]
```

```
##           name population  gdp     gdppc
## 3     Denmark         5.9  400  67.79661
## 4     Finland         5.5  300  54.54545
## 5     Germany        84.4 4456  52.79621
## 6     Holland        17.6  991  56.30682
## 1      Norway         5.5  593 107.81818
## 8      Poland        40.0 1801  45.02500
## 2      Sweden        10.7  593  55.42056
## 7 Switzerland         8.8  885 100.56818
```

3.5

a.

```
with(DF, plot(population, gdp, main = "Population and GDP",
              xlab = "Population",
              ylab = "GDP"))
```

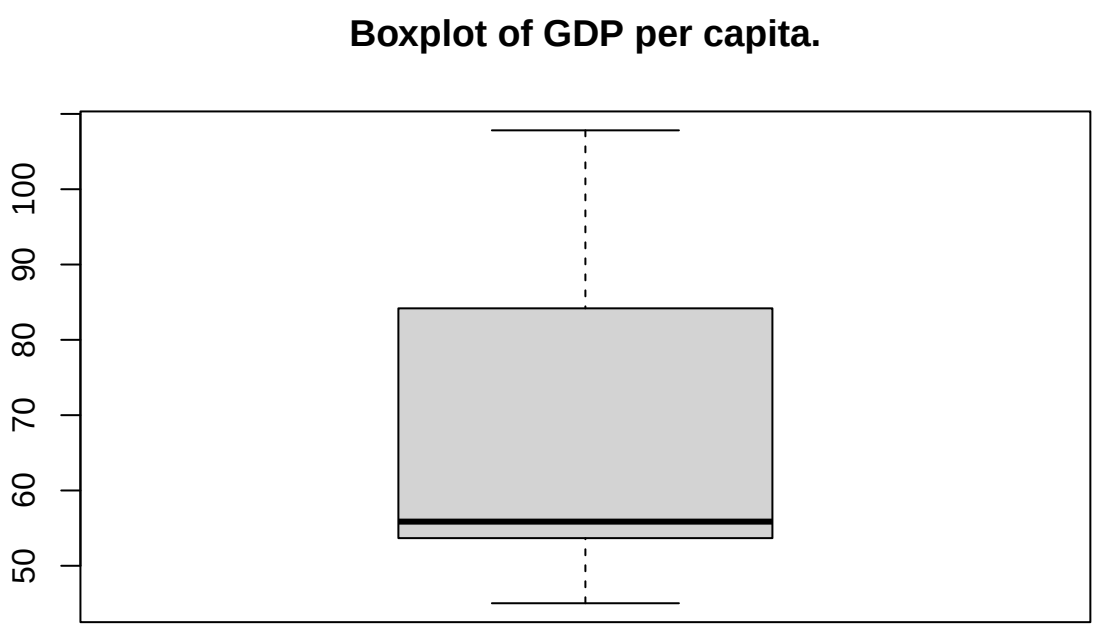


Alternatively, the code `plot(DF$population, DF$gdp, ...)` should do the same.

b. Rstudio actions in the "Plots" window.

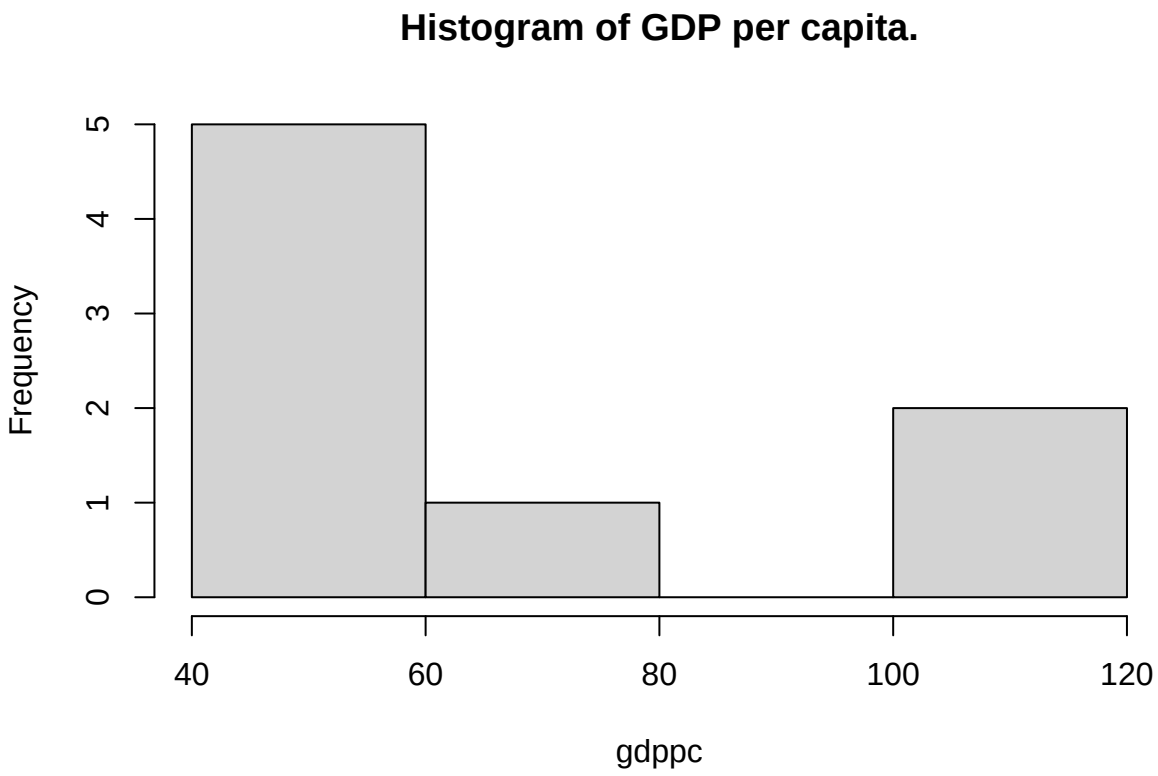
c. Boxplot for `gdppc`.

```
with(DF, boxplot(gdppc, main = "Boxplot of GDP per capita."))
```



d.

```
with(DF, hist(gdppc, main = "Histogram of GDP per capita."))
```



3.6

a.

```
DF[8,3] <- NA
DF
```

```
##          name population  gdp     gdppc
## 1      Norway        5.5  593 107.81818
## 2      Sweden       10.7  593  55.42056
## 3     Denmark        5.9  400  67.79661
## 4     Finland        5.5  300  54.54545
## 5     Germany       84.4 4456  52.79621
## 6     Holland       17.6  991  56.30682
## 7 Switzerland        8.8  885 100.56818
## 8      Poland       40.0   NA  45.02500
```

b.

```
mean(DF$gdp, na.rm = TRUE)
```

```
## [1] 1174
```

c.

```
summary(DF)
```

```
##      name              population         gdp              gdppc
##  Length:8           Min.   : 5.50   Min.   : 300.0   Min.   : 45.02
##  Class :character   1st Qu.: 5.80   1st Qu.: 496.5   1st Qu.: 54.11
##  Mode  :character   Median : 9.75   Median : 593.0   Median : 55.86
##                     Mean   :22.30   Mean   :1174.0   Mean   : 67.53
##                     3rd Qu.:23.20   3rd Qu.: 938.0   3rd Qu.: 75.99
##                     Max.   :84.40   Max.   :4456.0   Max.   :107.82
##                                     NA's   :1
```

We get a summary as usual, with the `na.rm` option active for means etc. In addition, `NA`'s are counted for each variable.

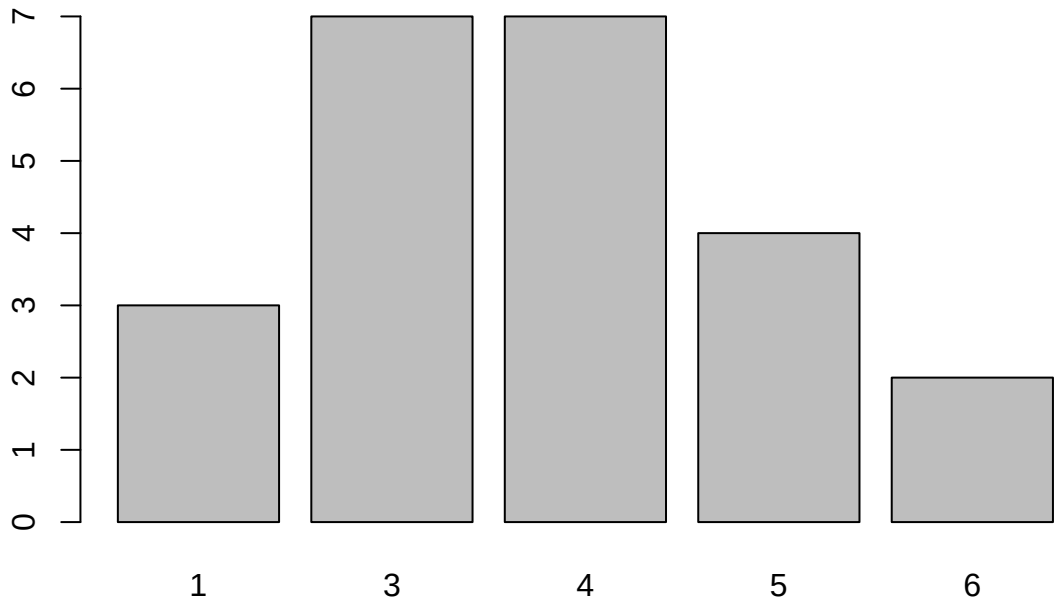
d.

```
toss <- c(4,4,3,5,3,6,3,5,4,3,4,3,4,1,3,1,5,5,4,3,4,1,6)

t <- table(toss)
t
```

```
## toss
## 1 3 4 5 6
## 3 7 7 4 2
```

```
barplot(t)
```

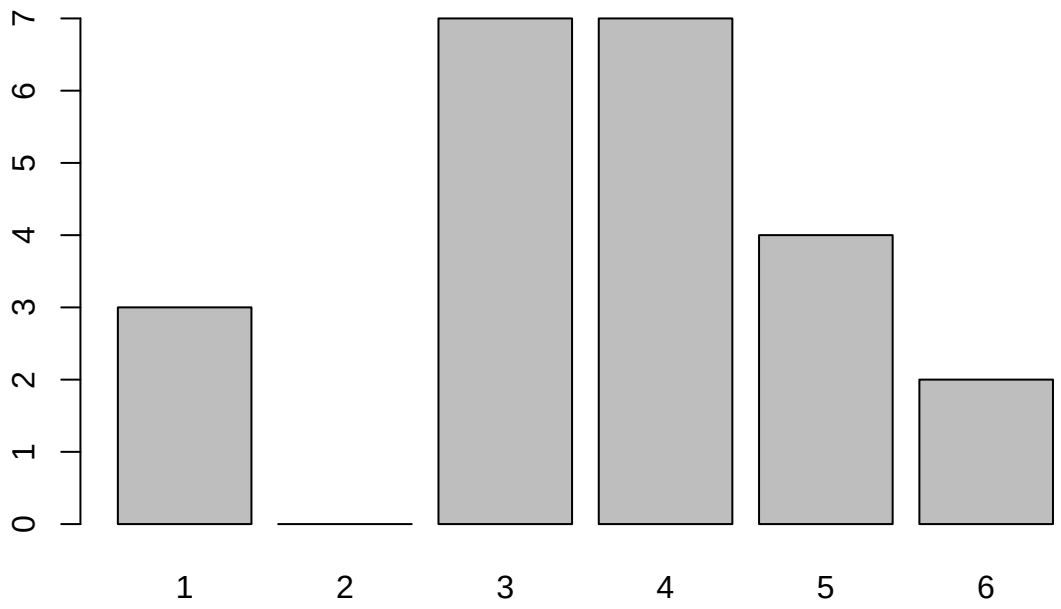


The problem is that `table` only counts the observed tosses, not knowing a priori that the posssible outcomes are 1 - 6. In the particular sequence, no 2's appear, so the result is correct but somewhat misleading. We should define toss as a `factor` to resolve the issue:

```
tossfac <- factor(toss, levels = 1:6)
t <- table(tossfac)
t
```

```
## tossfac
## 1 2 3 4 5 6
## 3 0 7 7 4 2
```

```
barplot(t)
```



That looks better.

e. Assuming a fair die, and independent tosses, the probability of *not* getting a "2" is 5/6. For this to happen 20 times in a row, we can obtain the probability by just multiplying the probability in each toss. Thus, since there are 20 tosses, we get

```
prob <- (5/6)^20
prob
```

```
## [1] 0.02608405
```

So, it happens in about 1 out of 40 such sequences. $(1/40 = 0.025)$

3.7

Activating the data:

```
AQ <- airquality
```

a. Getting main info for the dataframe.

```
nrow(AQ)
```

```
## [1] 153
```

```r
ncol(AQ)
```

```
## [1] 6
```

```r
names(AQ)
```

```
## [1] "Ozone"   "Solar.R" "Wind"    "Temp"    "Month"   "Day"
```

```r
head(AQ)
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
## 6    28      NA 14.9   66     5   6
```

b.

```r
names(AQ)[2] <- "SolarRad"
```

c.

```r
summary(AQ)
```

```
##      Ozone           SolarRad          Wind             Temp           Month
##  Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00   Min.   :5.000
##  1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:72.00   1st Qu.:6.000
##  Median : 31.50   Median :205.0   Median : 9.700   Median :79.00   Median :7.000
##  Mean   : 42.13   Mean   :185.9   Mean   : 9.958   Mean   :77.88   Mean   :6.993
##  3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00   3rd Qu.:8.000
##  Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00   Max.   :9.000
##  NA's   :37       NA's   :7
##       Day
##  Min.   : 1.0
##  1st Qu.: 8.0
##  Median :16.0
##  Mean   :15.8
##  3rd Qu.:23.0
##  Max.   :31.0
##
```

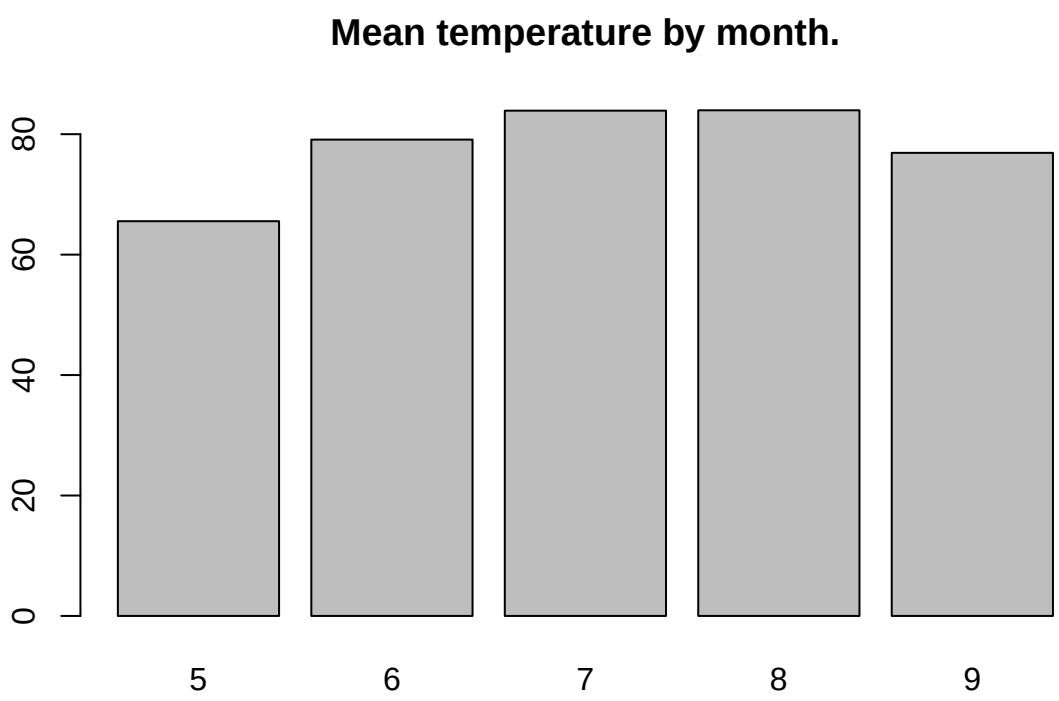
We find 37 NA's in `Ozone`, 7 in `SolarRad`.

d.

```r
mtemp <- with(AQ, tapply(Temp, Month, mean))
mtemp
```

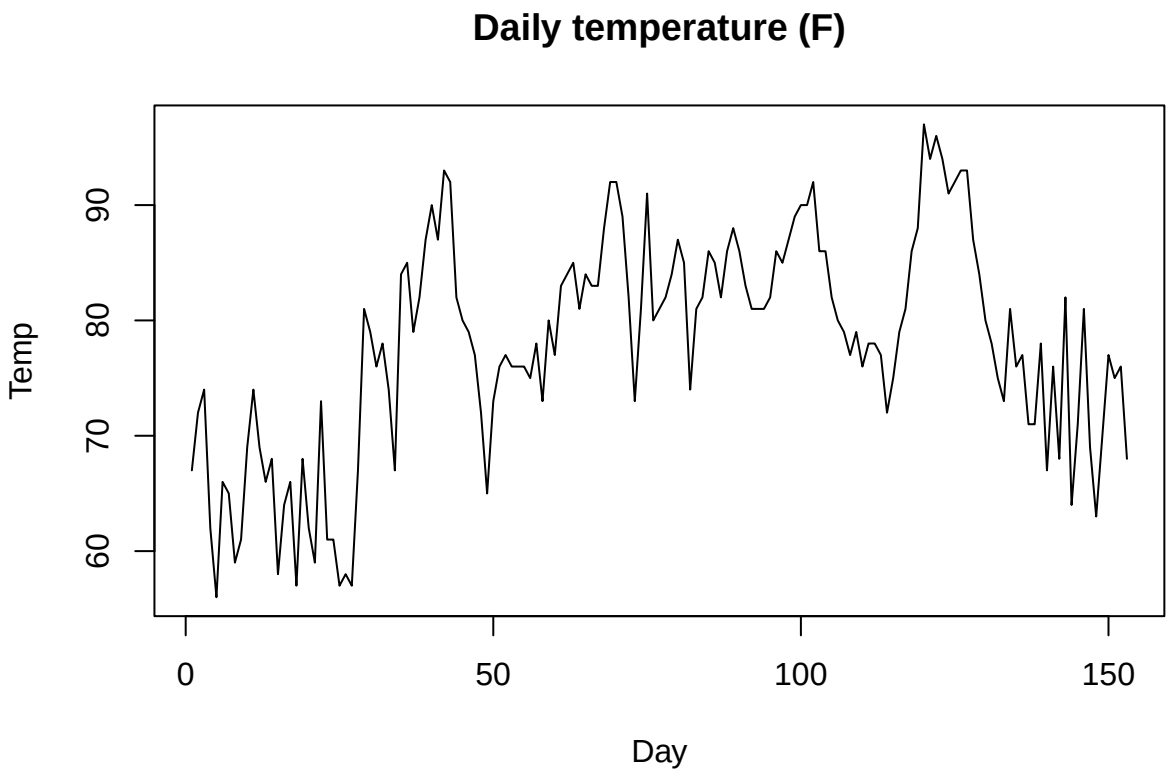
```
##        5        6        7        8        9
## 65.54839 79.10000 83.90323 83.96774 76.90000
```

```
barplot(mtemp, main="Mean temperature by month.")
```



We see highest mean temperatures in July and August.

```
with(AQ, plot(Temp,
              type = "l",
              xlab = "Day",
              main = "Daily temperature (F)"),
              lwd = 2)
```



We get what is usually called a "time series plot". One temperature observation per day is plotted, with time along the first axis.

f.

```
with(AQ, tapply(SolarRad, Month, mean))
```

```
##        5        6        7        8        9
##       NA 190.1667 216.4839       NA 167.4333
```

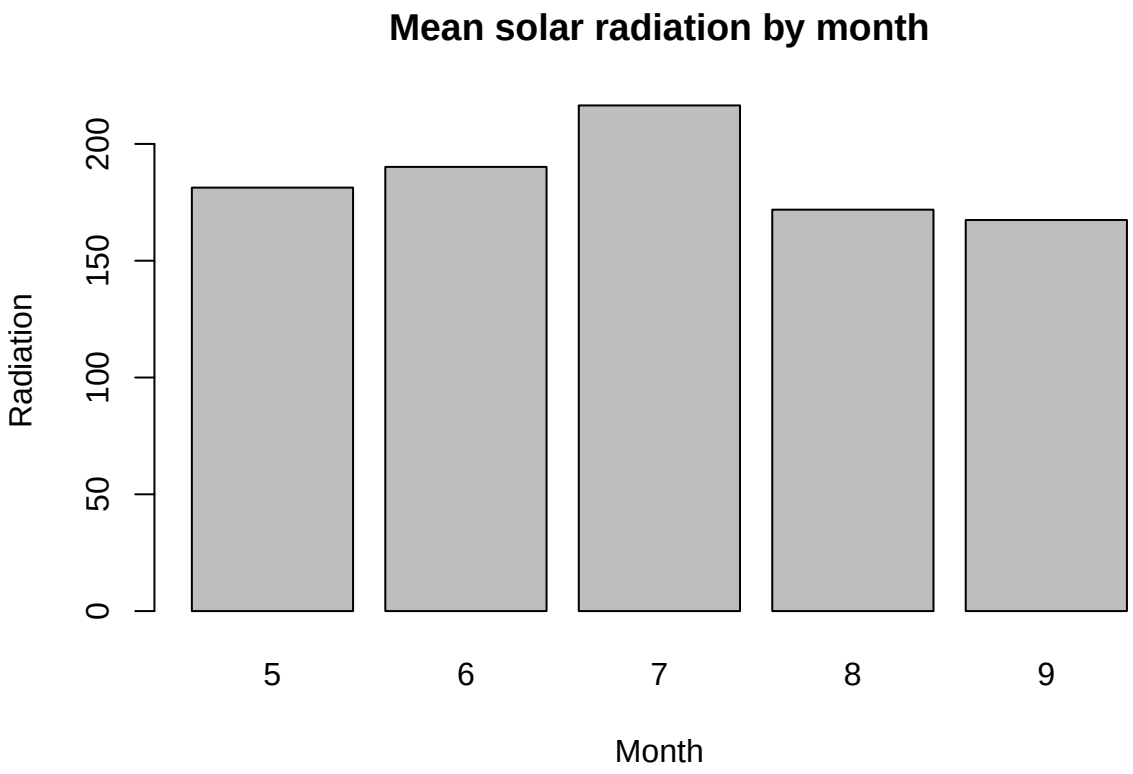
We can avoid getting NA results with this code:

```
with(AQ, tapply(SolarRad, Month, mean, na.rm=TRUE))
```

```
##        5        6        7        8        9
## 181.2963 190.1667 216.4839 171.8571 167.4333
```

g.

```
solradmean <- with(AQ, tapply(SolarRad, Month, mean, na.rm=TRUE))
barplot(solradmean, main = "Mean solar radiation by month",
                    xlab = "Month",
                    ylab = "Radiation")
```

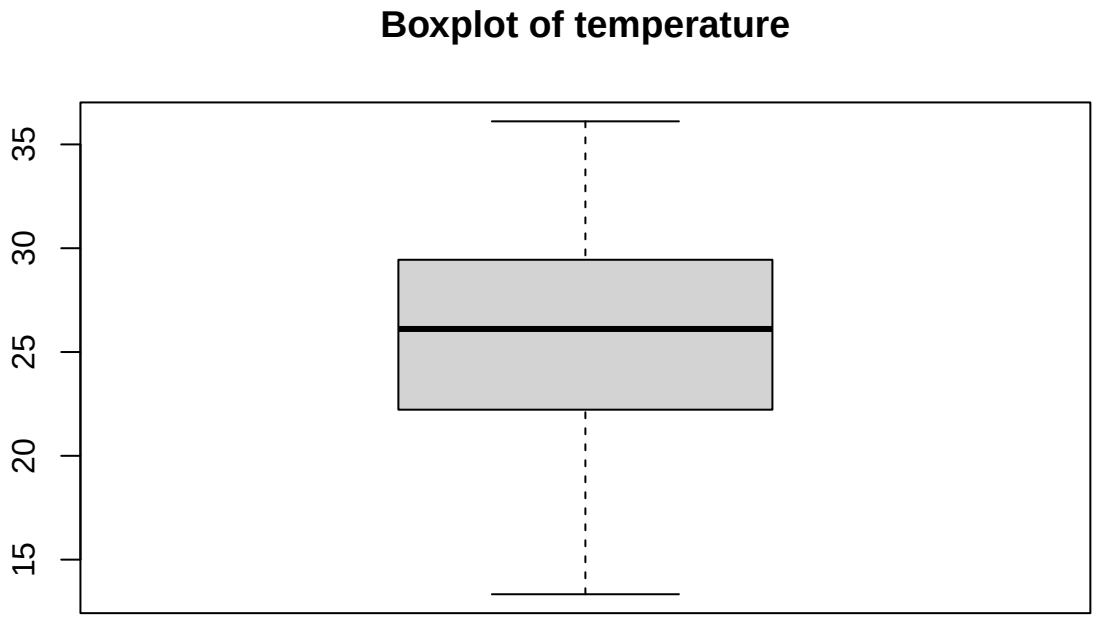


h. We need to invent a name for the new variable, e.g. `celsiustemp`.

```
AQ$celsiustemp <- (5/9)*(AQ$Temp - 32)
```

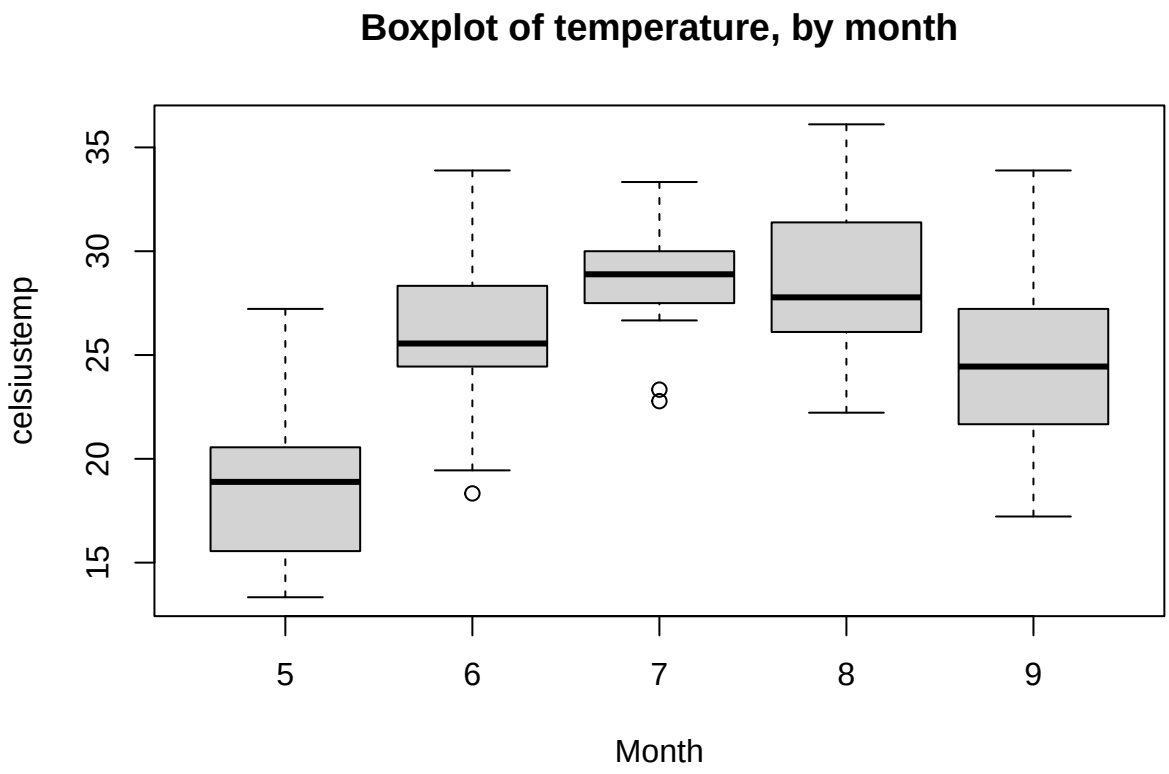
i. All data in one boxplot:

```
with(AQ, boxplot(celsiustemp, main = "Boxplot of temperature"))
```



Data grouped by month:

```
with(AQ, boxplot(celsiustemp ~ Month, main = "Boxplot of temperature, by month"))
```



j. Note here that it does not matter if we measure temperature by Fahrenheit or Celsius degrees. (why not?)

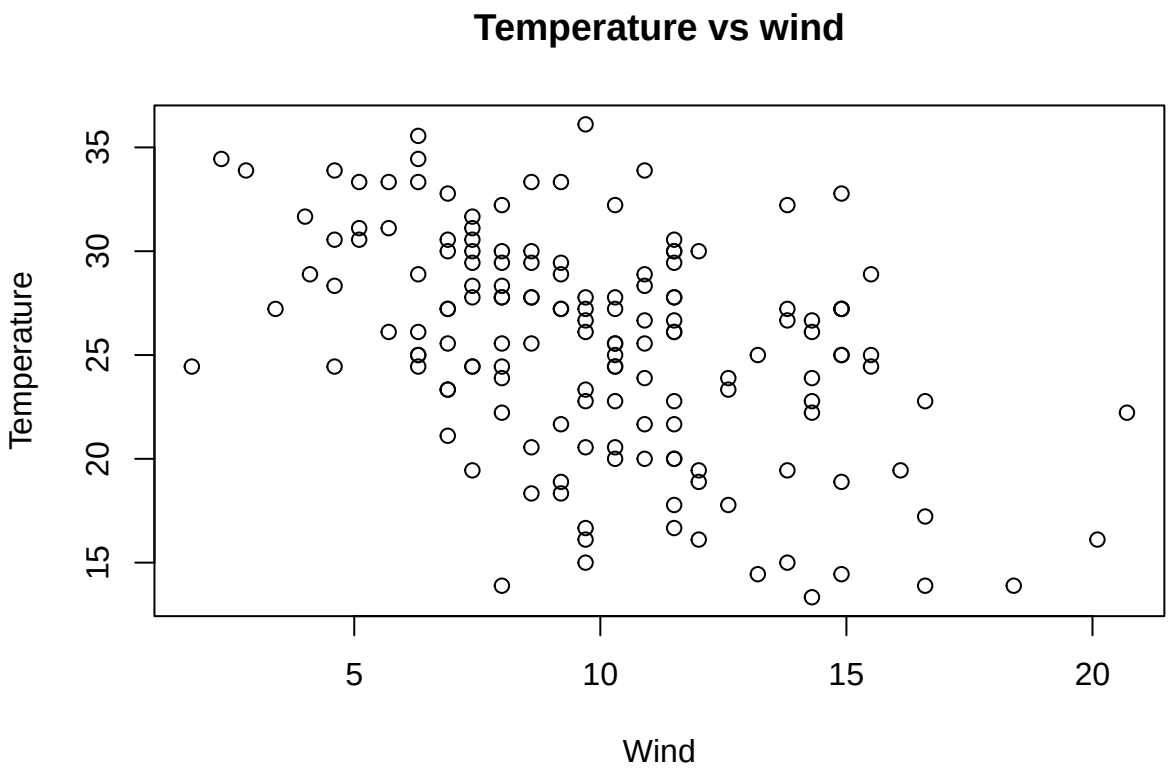
```
with(AQ, cor(Wind, Temp))
```

```
## [1] -0.4579879
```

```
with(AQ, cor(Wind, celsiustemp))
```

```
## [1] -0.4579879
```

```
with(AQ, plot(Wind, celsiustemp, main = "Temperature vs wind",
              ylab = "Temperature"))
```



We can spot the negative correlation in the plot. Finally we can get the whole correlation matrix as follows. In this case we may want to use only one of the temperature columns, e.g. the original one. That means we can select the first 6 columns of `AQ` and use the `cor` function on the resulting dataframe (i.e. the original `AQ`).

```
#get the 6 first columns:
AQorig <- AQ[ , 1:6]

#get the correlation matrix, where we do not use NA observations.
cor(AQorig, use = "complete.obs")
```

```
##                Ozone    SolarRad        Wind       Temp        Month          Day
## Ozone     1.000000000  0.34834169 -0.61249658  0.6985414  0.142885168 -0.005189769
## SolarRad  0.348341693  1.00000000 -0.12718345  0.2940876 -0.074066683 -0.057753801
## Wind     -0.612496576 -0.12718345  1.00000000 -0.4971897 -0.194495804  0.049871017
## Temp      0.698541410  0.29408764 -0.49718972  1.0000000  0.403971709 -0.096545800
## Month     0.142885168 -0.07406668 -0.19449580  0.4039717  1.000000000 -0.009001079
## Day      -0.005189769 -0.05775380  0.04987102 -0.0965458 -0.009001079  1.000000000
```

3.8

c. In my case, the code to read the file looks like below.

```
flights_NO <- read.csv("M:/Undervisning/Undervisningh24/Log708/Data/flights_NO.csv")
head(flights_NO)
```

```
##   Origin.Code Destination.Code Dep.Time Arr.Time Flight Airline.Code      Alliance
## 1         AES              BGO      715      800   4139           SK Star Alliance
## 2         AES              BGO      715      800   4139           SK Star Alliance
## 3         AES              BGO      715      800   4139           SK Star Alliance
## 4         AES              BGO      715      800   4139           SK Star Alliance
## 5         AES              BGO      715      800   4139           SK Star Alliance
## 6         AES              BGO     1100     1140   4147           SK Star Alliance
##   Origin.Country Destination.Country Day.of.Week Block.Mins Seats
## 1         Norway              Norway      Monday         45   181
## 2         Norway              Norway     Tuesday         45   141
## 3         Norway              Norway   Wednesday         45   141
## 4         Norway              Norway    Thursday         45   141
## 5         Norway              Norway      Friday         45   141
## 6         Norway              Norway      Monday         40    90
```

d.

```
df2 <- subset(flights_NO, Day.of.Week == "Monday" & Seats > 50)

nrow(flights_NO)
```

```
## [1] 4897
```

```
nrow(df2)
```

```
## [1] 488
```

The dataframe `df2` contains the data for flights going on Mondays, with airplanes having more than 50 seats.

e.

```
write.csv(df2, "LargeMondayFlights.csv")

dir()
```

```
##  [1] "_Ch3Solutions.Rmd"      "Ch1Solutions.html"      "Ch1Solutions.pdf"
##  [4] "Ch1Solutions.Rmd"       "Ch2Solutions.html"      "Ch2Solutions.pdf"
##  [7] "Ch2Solutions.Rmd"       "Ch3Solutions.html"      "Ch3Solutions.pdf"
## [10] "Ch3Solutions.Rmd"       "Ch3Solutions_files"     "Ch3Solutions_full.html"
## [13] "Ch4Solutions.html"      "Ch4Solutions.pdf"       "Ch4Solutions.Rmd"
## [16] "Ch5Solutions.html"      "Ch5Solutions.pdf"       "Ch5Solutions.Rmd"
## [19] "Ch6Solutions.html"      "Ch6Solutions.log"       "Ch6Solutions.pdf"
## [22] "Ch6Solutions.Rmd"       "Ch7Solutions.html"      "Ch7Solutions.pdf"
## [25] "Ch7Solutions.Rmd"       "Ch8Solutions.html"      "Ch8Solutions.pdf"
```

```
## [28] "Ch8Solutions.Rmd"       "friendsfile.csv"        "LargeMondayFlights.csv"
```

We can see the file in the working directory.

f.

```
df3 <- read.csv("LargeMondayFlights.csv")

identical(df2, df3)
```

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

g. We just have to provide the whole file path:

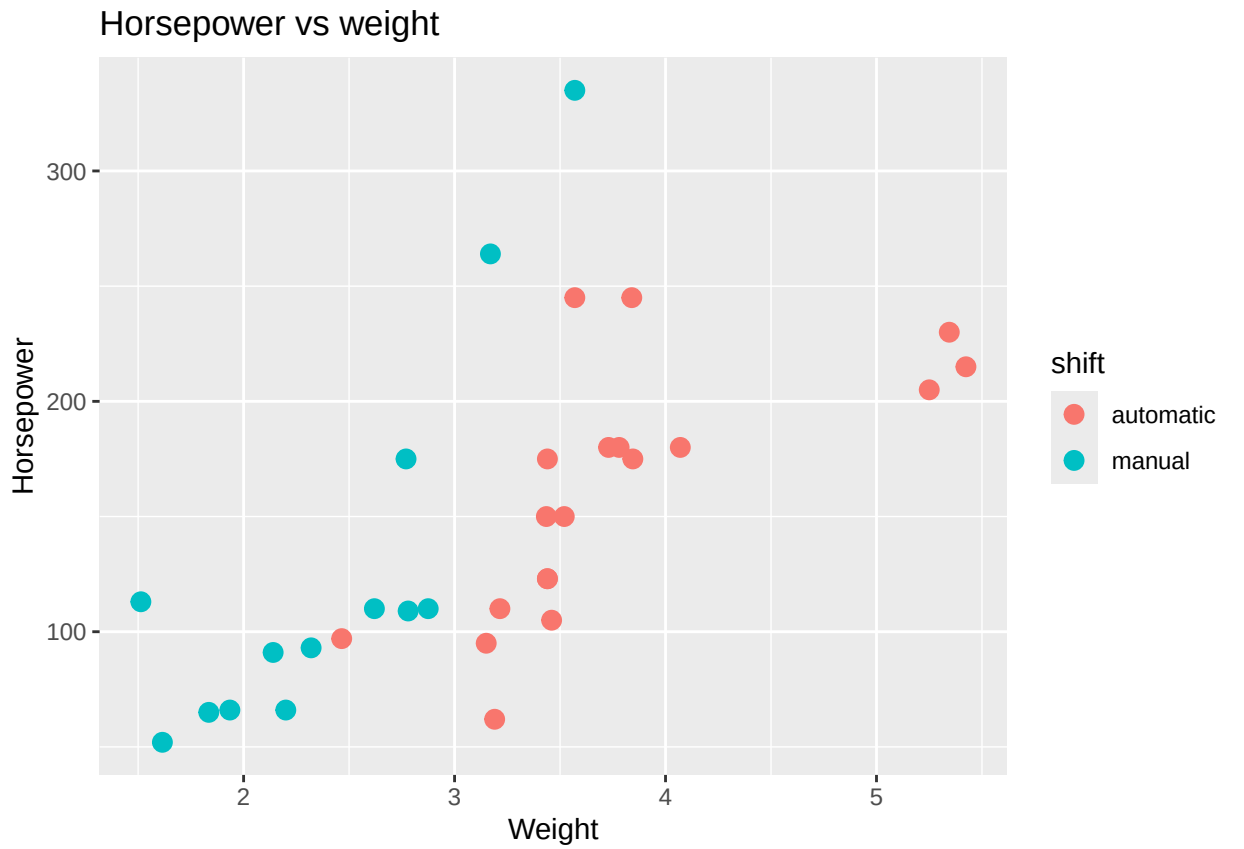
```
write.csv(df2, "M:/Undervisning/Undervisningh24/Log708/Data/LargeMondayFlights.csv")
```

3.9

```
library(ggplot2)

#Redefine variable "am" as a factor, just to make figure labels look right.
shift <- factor(mtcars$am, labels = c("automatic", "manual"))

#make a plot
ggplot(mtcars, aes(x = wt, y = hp, color = shift)) +
  geom_point(size = 3) +
  labs(title = "Horsepower vs weight",
       x = "Weight",
       y = "Horsepower")
```

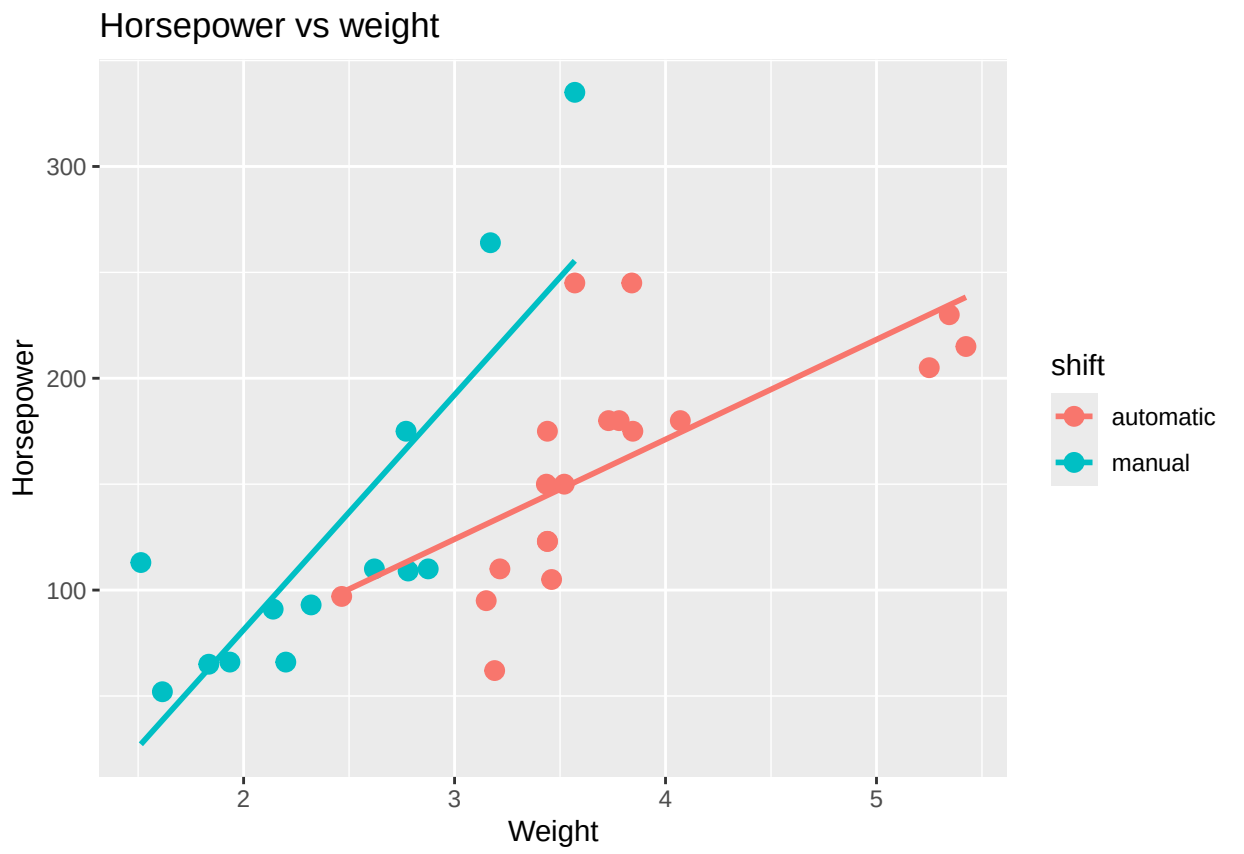


a. Horsepower is positively correlated to weight, as expected. Automatic shift is mainly present in heavier vehicles in these data.

b. The `+ geom_smooth(method = lm, se = FALSE)` code added gives the same figure with regression lines for each group defined by the `shift` variable.

```
#make a plot
ggplot(mtcars, aes(x = wt, y = hp, color = shift)) +
  geom_point(size = 3) +
  labs(title = "Horsepower vs weight",
       x = "Weight",
       y = "Horsepower") +
       geom_smooth(method = lm, se = FALSE)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



3.10

a. We can use `?Normal` with capital "N". Or, one can use `?distributions` and then find that the normal distributions can be found under `dnorm`.
Then `?dnorm` will give the desired information. Here we can learn that there are four variants, among which `pnorm` gives cumulative probabilities. So then we can find the probabilities in the question as

```
pnorm(2.1)
```

```
## [1] 0.9821356
```

```
(1 - pnorm(2.1))
```

```
## [1] 0.01786442
```

```
# or:
pnorm(2.1, lower.tail = FALSE)
```

```
## [1] 0.01786442
```

```
pnorm(12, mean = 10, sd = 3)
```

```
## [1] 0.7475075
```

```
pnorm(10.9, mean = 10, sd = 3) - pnorm(8.4, mean = 10, sd = 3)
```

```
## [1] 0.32101
```

b. A Google search for "R calculate normal distribution" leads (among many other hits) to the web page at Tutorialspoint.

c. Reading carefully the internal or external information tells us we need to use the function `qnorm` to answer this question. So, to find b as specified in the question we need to do

```
qnorm(0.80)
```

```
## [1] 0.8416212
```

We also note that the answer "looks right" as we are working on the standard normal distribution here. We could also check with the table in chapter 2.

d. This is similar to the question above, but using a general normal distribution. Obviously, the number c must satisfy $P[X \leq c] = 0.10$ (law of complement), so we can do

```
qnorm(0.10, mean = 10, sd = 3)
```

```
## [1] 6.155345
```

Alternatively using the option `lower.tail = FALSE` we can write

```
qnorm(0.90, mean = 10, sd = 3, lower.tail = FALSE)
```

```
## [1] 6.155345
```

e. From the documentation (and as shown in chapter 3) this is done via

```
set.seed(3333)
my_x <- rnorm(30, mean = 10, sd = 3)

#compute mean and standard deviation
M <- mean(my_x)
S <- sd(my_x)

#show result:
c(M, S)
```

```
## [1] 10.720201  2.497178
```

We see the numbers fairly close to the "true" parameter values 10 and 3.