Computing in R: exercises

Perry Moerland

March 11-15, 2024

The objective of these computer exercises is to become familiar with the basic structure of R and to learn some more sophisticated tips and tricks of R. The R code to perform the analyses, together with the output, is given in a separate file. However, you are strongly advised to write the code yourself. Questions are posed to reflect on what you are doing and to study output and results in more detail.

1 Basics

If you have not done so already, start R via **Starten - Alle programma's - R - R 4.3.3** (older versions of R should also be fine). This opens the R console. R is a command line driven environment. This means that you have to type in commands (line-by-line) for it to compute or calculate something. In its simplest form R can therefore be used as a pocket calculator:

```
2+2
```

```
# [1] 4
```

Question 1. **(a)** Look up today's exchange rate of the euro versus the US$ on the Internet. Use R to calculate how many US$ is 15 euro.

Answer

```
# This is more or less the current exchange rate
15 * 1.094
```

```
# [1] 16.41
```

```
# It is better to use a variable to store the exchange rate
exrate <- 1.094
exrate
```

```
# [1] 1.094
```

(b) Round the number you found to the second decimal using function `round` (use `help` or `?` to inspect the arguments of the function) and assign the result to an object `curr`.

Answer

```
curr <- round(15 * exrate, digits = 2)
curr
```

```
# [1] 16.41
```

(c) Use `mode`, `str`, and `summary` to inspect the object you created. These functions give a compact display of the type of object and its contents.

Answer

```
mode(curr)

# [1] "numeric"
str(curr)

#  num 16.4
summary(curr)

#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#   16.41   16.41   16.41   16.41   16.41   16.41
```

Luckily R is more than just a calculator, it is a programming language with most of the elements that every programming language has: statements, objects, types, classes *etc.*

2 R syntax: vectors

Question 2. **(a)** One of the simplest data structures in R is a vector. Use the function `seq` to generate a vector `vec` that consists of all numbers from 11 to 30 (see `?seq` for documentation).

Answer

```
vec <- seq(11,30)
vec

#  [1] 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
```

(b) Select the 7th element of the vector.

Answer

```
vec[7]

# [1] 17
```

(c) Select all elements except the 15th.

Answer

```
# Use '-' to exclude elements
vec[-15]

#  [1] 11 12 13 14 15 16 17 18 19 20 21 22 23 24 26 27 28 29 30
```

(d) Select the 2nd and 5th element of the vector.

Answer

```
# Use the concatenate function 'c'
vec[c(2,5)]

# [1] 12 15
```

(e) Select only the odd valued elements of the vector `vec`. Do this is in two steps: first create the appropriate vector of indices `index` using the function `seq`, then use `index` to make the selection.

Answer

```
# Instead of "hard coding" the length of the vector by specifying the value 20,
# you can calculate it using the function 'length'. This makes your code more
# generic and less error prone
index <- seq(1,length(vec),by=2)
vec[index]
```

```
#  [1] 11 13 15 17 19 21 23 25 27 29
```

Until now you have only used the R console. However, RStudio (**Starten - Alle programma's - R - RStudio**) provides a much nicer and richer interface when working with R. Although all exercises can be made within the basic R environment, we highly recommend to use RStudio.

Question 3. **(a)** What are the windows that RStudio consists of?

Answer

The most important ones are Script, Console, Environment, History, Files, Plots, Packages, Help, Viewer.

From now on we advise you to use RStudio. You can use either the Console or the Script window (the upper left window). We recommend you to use the Script window, since this allows you to easily save your code to a file for later usage.

(b) Use the elementary functions `/`, `-`, `^` and the functions `sum` and `length` to calculate the mean $\bar{x} = \sum_i x_i/n$ and the standard deviation $\sqrt{\sum_i (x_i - \bar{x})^2/(n-1)}$ of the vector `vec` of all numbers from 11 to 30. You can verify your answer by using the built-in R functions `mean` and `sd`.

Answer

```
# Don't call the variable 'mean' since the function 'mean' already exists
xmean <- sum(vec)/length(vec)
xmean
```

```
# [1] 20.5
```

```
mean(vec)
```

```
# [1] 20.5
```

```
# Calculate the standard deviation in three (baby) steps
numerator <- sum((vec-xmean)^2)
denominator <- (length(vec)-1)
xstd  <- sqrt(numerator/denominator)
xstd
```

```
# [1] 5.91608
```

```
sd(vec)
```

```
# [1] 5.91608
```

(c) Once you completed the analysis, have a look at the Environment and History windows. Do you understand their contents?

Question 4. R comes with many predefined datasets. You can type `data()` to get the whole list. The `islands` dataset gives the areas of the world's major landmasses exceeding 10,000 square miles. It can be loaded by typing:

```
# This could in this case actually be skipped since the package datasets is
# already loaded
data(islands)
```

(a) Inspect the object using the functions `head`, `str` *etc.* Also have a look at the help file using `help(islands)`.

(b) How many landmasses in the world exceeding 10,000 square miles are there?

Answer

```
# From ?islands: "Description: The areas in thousands of square miles of
# the landmasses which exceed 10,000 square miles." So all of them:
length(islands)
```

```
# [1] 48
```

(c) Make a Boolean vector that has the value `TRUE` for all landmasses exceeding 20,000 square miles.

Answer

```
# Remember that area was given in thousands of square miles
islands.more20 <- (islands > 20)
```

(d) Select the islands with landmasses exceeding 20,000 square miles.

Answer

```
# Use the Boolean vector islands.more20 to select the ones with value TRUE
islands[islands.more20]
```

```
#          Africa      Antarctica            Asia       Australia          Baffin
#           11506            5500           16988            2968             184
#           Banks          Borneo         Britain         Celebes           Celon
#              23             280              84              73              25
#            Cuba           Devon       Ellesmere          Europe       Greenland
#              43              21              82            3745             840
#      Hispaniola        Hokkaido          Honshu         Iceland         Ireland
#              30              30              89              40              33
#            Java           Luzon      Madagascar        Mindanao        Moluccas
#              49              42             227              36              29
#      New Guinea New Zealand (N) New Zealand (S)    Newfoundland   North America
#             306              44              58              43            9390
#   Novaya Zemlya        Sakhalin   South America         Sumatra        Tasmania
#              32              29            6795             183              26
#        Victoria
#              82
```

(e) Make a character vector that only contains the names of the islands.

Answer

```
islands.names <- names(islands)
```

(f) The Moluccas have mistakenly been counted as a single island. The largest island of the Moluccas, Halmahera, only has about 7,000 square miles. Remove Moluccas from the data. Hint: an elegant solution uses the Boolean operator `!=`.

Answer

```
notMoluccas <- islands.names != "Moluccas"
islands.withoutMoluccas <- islands[notMoluccas]
# Another solution is:
islands.withoutMoluccas <- subset(islands,islands.names!="Moluccas")
```

3 R syntax: matrices

Question 5. **(a)** Create a character matrix `M` with 6 rows and three columns containing the first eighteen letters of the Roman alphabet (hint: see `?letters`). Fill the matrix column-wise.

Answer

```
letters

#  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
# [20] "t" "u" "v" "w" "x" "y" "z"

M <- matrix(letters[1:18], nrow = 6, ncol = 3)
M

#      [,1] [,2] [,3]
# [1,] "a"  "g"  "m"
# [2,] "b"  "h"  "n"
# [3,] "c"  "i"  "o"
# [4,] "d"  "j"  "p"
# [5,] "e"  "k"  "q"
# [6,] "f"  "l"  "r"
```

(b) Which is the letter on the fifth row and second column?

Answer

```
M[5, 2]

# [1] "k"
```

(c) Replace the letters in the last column of `M` with the last six letters of the Roman alphabet.

Answer

```
M[,3] <- letters[21:26]
M

#      [,1] [,2] [,3]
# [1,] "a"  "g"  "u"
# [2,] "b"  "h"  "v"
# [3,] "c"  "i"  "w"
# [4,] "d"  "j"  "x"
# [5,] "e"  "k"  "y"
# [6,] "f"  "l"  "z"
```

(d) Make a character vector containing all the vowels and use this vector to replace the vowels in `M` with the character string "vowel".

Answer

```r
# Let's assume that "y" is also a vowel
vowels <- c("a","e","i","o","u","y")
M[M %in% vowels] <- "vowel"
M
```

```
#      [,1]    [,2]    [,3]
# [1,] "vowel" "g"     "vowel"
# [2,] "b"     "h"     "v"
# [3,] "c"     "vowel" "w"
# [4,] "d"     "j"     "x"
# [5,] "vowel" "k"     "vowel"
# [6,] "f"     "l"     "z"
```

(e) Now replace all consonants in `M` with the character string "consonant".

Answer

```r
# '!=': not equal to
M[M!="vowel"] <- "consonant"
M
```

```
#      [,1]        [,2]        [,3]
# [1,] "vowel"     "consonant" "vowel"
# [2,] "consonant" "consonant" "consonant"
# [3,] "consonant" "vowel"     "consonant"
# [4,] "consonant" "consonant" "consonant"
# [5,] "vowel"     "consonant" "vowel"
# [6,] "consonant" "consonant" "consonant"
```

(f) Count the number of elements in `M` that are equal to "consonant" and "vowel" respectively.

Answer

```r
sum(M=="consonant")
```

```
# [1] 13
```

```r
sum(M=="vowel")
```

```
# [1] 5
```

4 Data frames, data import, missing values and factors

Question 6. We are going to use yet another predefined dataset: `mtcars`. These data were extracted from the 1974 Motor Trend US magazine, and comprise fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models). See `?mtcars` for the definition of the variables contained in it.

(a) What kind of data structure is the `mtcars` data?

Answer

```
# See '?mtcars', it is a data frame. On the command line you can use 'class' or 'str'
class(mtcars)
```

```
# [1] "data.frame"
```

```
str(mtcars)
```

```
# 'data.frame': 32 obs. of  11 variables:
#  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
#  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
#  $ disp: num  160 160 108 258 360 ...
#  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
#  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
#  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
#  $ qsec: num  16.5 17 18.6 19.4 17 ...
#  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
#  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
#  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
#  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

(b) How many car models can ride more than 25 miles per gallon?

Answer

```
sum(mtcars$mpg > 25)
```

```
# [1] 6
```

```
# Other solutions using different ways of extracting a column from a data frame
sum(mtcars[,"mpg"] > 25)
```

```
# [1] 6
```

```
sum(mtcars["mpg"] > 25)
```

```
# [1] 6
```

```
sum(mtcars[,1] > 25)
```

```
# [1] 6
```

(c) What are the names of the car models that can ride more than 25 miles per gallon?

Answer

```
rownames(mtcars[mtcars$mpg > 25,])
```

```
# [1] "Fiat 128"       "Honda Civic"    "Toyota Corolla" "Fiat X1-9"
# [5] "Porsche 914-2"  "Lotus Europa"
```

(d) Select the car models that can ride less than 22 miles per gallon and have more than 6 cylinders.

Answer

```
# Use the Boolean operator '&' (logical AND)
mtcars[(mtcars$mpg < 22) & (mtcars$cyl > 6), ]

#                      mpg  cyl  disp  hp drat    wt  qsec vs am gear carb
# Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
# Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
# Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
# Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
# Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
# Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
# Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
# Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
# Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
# AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
# Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
# Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
# Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
# Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
```

(e) In addition to the previous selection, also select the car models that either have more than 3 carburetors or weigh less than 3500 lbs. How many car models satisfy these conditions?

Answer

```
# Use the Boolean operator '|' (logical OR).
# Note that the weight is specified per 1000 lbs (see ?mtcars)
mtcars[(mtcars$mpg < 22) & (mtcars$cyl > 6) & ((mtcars$carb > 3) | (mtcars$wt) < 3.5), ]

#                      mpg  cyl disp  hp drat    wt  qsec vs am gear carb
# Hornet Sportabout   18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
# Duster 360          14.3   8  360 245 3.21 3.570 15.84  0  0    3    4
# Cadillac Fleetwood  10.4   8  472 205 2.93 5.250 17.98  0  0    3    4
# Lincoln Continental 10.4   8  460 215 3.00 5.424 17.82  0  0    3    4
# Chrysler Imperial   14.7   8  440 230 3.23 5.345 17.42  0  0    3    4
# AMC Javelin         15.2   8  304 150 3.15 3.435 17.30  0  0    3    2
# Camaro Z28          13.3   8  350 245 3.73 3.840 15.41  0  0    3    4
# Ford Pantera L      15.8   8  351 264 4.22 3.170 14.50  0  1    5    4
# Maserati Bora       15.0   8  301 335 3.54 3.570 14.60  0  1    5    8
# The number of car models is the number of rows, that is the first dimension
dim(mtcars[(mtcars$mpg < 22) & (mtcars$cyl > 6) & ((mtcars$carb > 3) | (mtcars$wt) < 3.5), ])

# [1]  9 11
# The number of rows can also be retrieved with 'nrow'
nrow(mtcars[(mtcars$mpg < 22) & (mtcars$cyl > 6) & ((mtcars$carb > 3) | (mtcars$wt) < 3.5), ])

# [1] 9
```

(f) The variable `am` is a numeric variable that indicates the type of transmission:

- 0: automatic
- 1: manual

Add a variable called `am_char` to the `mtcars` data set where you recode the variable `am` and replace 0 by “automatic” and 1 by “manual”.

Answer

```
mtcars$am_char[mtcars$am == 0] <- "automatic"
mtcars$am_char[mtcars$am == 1] <- "manual"
mtcars[,c("am","am_char")]
```

```
#                      am   am_char
# Mazda RX4             1    manual
# Mazda RX4 Wag         1    manual
# Datsun 710            1    manual
# Hornet 4 Drive        0 automatic
# Hornet Sportabout     0 automatic
# Valiant               0 automatic
# Duster 360            0 automatic
# Merc 240D             0 automatic
# Merc 230              0 automatic
# Merc 280              0 automatic
# Merc 280C             0 automatic
# Merc 450SE            0 automatic
# Merc 450SL            0 automatic
# Merc 450SLC           0 automatic
# Cadillac Fleetwood    0 automatic
# Lincoln Continental   0 automatic
# Chrysler Imperial     0 automatic
# Fiat 128              1    manual
# Honda Civic           1    manual
# Toyota Corolla        1    manual
# Toyota Corona         0 automatic
# Dodge Challenger      0 automatic
# AMC Javelin           0 automatic
# Camaro Z28            0 automatic
# Pontiac Firebird      0 automatic
# Fiat X1-9             1    manual
# Porsche 914-2         1    manual
# Lotus Europa          1    manual
# Ford Pantera L        1    manual
# Ferrari Dino          1    manual
# Maserati Bora         1    manual
# Volvo 142E            1    manual
```

We will work with a data set that gives the survival status of passengers on the Titanic. See titanic3info.txt for a description of the variables. Some background information on the data can be found on titanic.html.

Question 7. **(a)** Download the `titanic` data set `titanic3.dta` in STATA format from the course website and import it into R using the function `read.dta`. Do not use the function `read_dta` from the **haven** package nor the ‘Import Dataset’ option in RStudio. This leads to slight differences in the imported data which may lead to problems in some of the exercises. Give the data set an appropriate name as an R object. E.g., we can call it `titanic3` (but feel free to give it another name). Before importing the data, you first have to load the **foreign** package in which the function `read.dta` is defined. Moreover, you probably will need to point R to the right folder where you saved the file `titanic3.dta`. You can do this by changing the so-called *working directory*, which will be explained later. Using the basic R environment you can do this via the menu (**File - Change dir**), and similarly for RStudio (**Session - Set Working Directory - Choose Directory**).

Answer

```
library(foreign)
titanic3 <- read.dta("titanic3.dta", convert.underscore=TRUE)
```

We take a look at the values in the `titanic` data set.

(b) First, make the whole data set visible in a spreadsheet like format (how this is done depends on whether base R or RStudio is used).

Answer

This can be done using `View` on the command line. You can also use the menu: in R use **Edit - Data editor**, in RStudio click on the name of the data set in the **Environment** window.

(c) Often, it is sufficient to show a few records of the total data set. This can be done via selections on the rows, but another option is to use the functions `head` and `tail`. Use these functions to inspect the first 4 and last 4 records (instead of 6 which is the default).

Answer

```
head(titanic3, n = 4)
```

```
#   pclass survived                            name    sex     age sibsp parch
# 1    1st        1   Allen, Miss. Elisabeth Walton female 29.0000     0     0
# 2    1st        1  Allison, Master. Hudson Trevor   male  0.9167     1     2
# 3    1st        0    Allison, Miss. Helen Loraine female  2.0000     1     2
# 4    1st        0 Allison, Mr. Hudson Joshua Crei   male 30.0000     1     2
#   ticket     fare   cabin    embarked boat body                       home.dest
# 1  24160 211.3375      B5 Southampton    2   NA                    St Louis, MO
# 2 113781 151.5500 C22 C26 Southampton   11   NA Montreal, PQ / Chesterville, ON
# 3 113781 151.5500 C22 C26 Southampton        NA Montreal, PQ / Chesterville, ON
# 4 113781 151.5500 C22 C26 Southampton       135 Montreal, PQ / Chesterville, ON
#         dob family  agecat
# 1 -28019.25     no (18,30]
# 2 -17761.82    yes  (0,18]
# 3 -18157.50    yes  (0,18]
# 4 -28384.50    yes (18,30]
```

```
tail(titanic3, n = 4)
```

```
#      pclass survived                      name    sex  age sibsp parch ticket
# 1306    3rd        0     Zabour, Miss. Thamine female   NA     1     0   2665
# 1307    3rd        0 Zakarian, Mr. Mapriededer   male 26.5     0     0   2656
# 1308    3rd        0       Zakarian, Mr. Ortin   male 27.0     0     0   2670
# 1309    3rd        0        Zimmerman, Mr. Leo   male 29.0     0     0 315082
#         fare cabin    embarked boat body home.dest       dob family  agecat
# 1306 14.4542         Cherbourg        NA                  NA    yes    <NA>
# 1307  7.2250         Cherbourg       304           -27106.12     no (18,30]
# 1308  7.2250         Cherbourg        NA           -27288.75     no (18,30]
# 1309  7.8750       Southampton        NA           -28019.25     no (18,30]
```

The function `dim` can be used to find the number of rows (passengers in this case) and columns (variables) in the data.

```
dim(titanic3)
```

An alternative is the function `str`. This function shows information per column (type of variable, first records, levels in case of factor variables) as well as the total number of rows and columns.

```
str(titanic3)
```

(d) Issue both commands and study the output.

Answer

```
dim(titanic3)
```

```
# [1] 1309   17
```

(e) Summarize the data set by using the `summary` function.

Answer

```
# Here the summary is shown for four of the variables only
summary(titanic3[,c("survived","pclass","home.dest","dob")])
```

```
#     survived     pclass     home.dest              dob
#  Min.   :0.000   1st:323   Length:1309        Min.   :-46647
#  1st Qu.:0.000   2nd:277   Class :character   1st Qu.:-31672
#  Median :0.000   3rd:709   Mode  :character   Median :-27654
#  Mean   :0.382                                Mean   :-28341
#  3rd Qu.:1.000                                3rd Qu.:-25097
#  Max.   :1.000                                Max.   :-17488
#                                               NA's   :263
```

This gives a summary of all the variables in the data set. We can see that for categorical variables like `pclass` a frequency distribution is given, whereas for continuous variables like `age` numerical summaries are given. Still, for some variables we do not automatically get what we would like or expect:

- The variable `survived` is dichotomous with values zero and one, which are interpreted as numbers.
- The categorical variable `pclass` is represented differently from the categorical variable `home.dest`.
- The variable `dob`, which gives the date of birth, is represented by large negative numbers.

In subsequent exercises, we will shed further light on these anomalies and we will try to repair some of these.

Question 8. We can also summarize specific columns (variables) of a `data.frame`. There are many ways to summarize a variable, depending on its structure and variability. For continuous variables, the same `summary` function can be used, but other options are the functions `mean`, `quantile`, `IQR`, `sd` and `var`. For categorical summaries, one may use `summary` and `table`. Note that missing values are treated differently depending on the function used.

(a) Summarize the age variable in the `titanic` data set (the age column is selected via `titanic3$age`). Give the 5%, 25%, 50%, 75% and 95% quantiles and the inter-quartile range. You may have to take a look at the help files for the functions.

Answer

```
# Note that you have to convert percentages into probabilities
quantile(titanic3$age, probs=c(0.05,0.25,0.5,0.75,0.95), na.rm=TRUE)
```

```
#  5% 25% 50% 75% 95%
#   5  21  28  39  57
```

```
IQR(titanic3$age, na.rm=TRUE)
```

```
# [1] 18
```

(b) Summarize the `survived` variable using the `table` function. Also give a two-by-two table for sex and survival status using the same `table` function.

Answer

```
table(titanic3$survived)
```

```
#
#   0   1
# 809 500
```

```
table(titanic3$sex, titanic3$survived)
```

```
#
#            0   1
#   female 127 339
#   male   682 161
```

Question 9 (OPTIONAL).

Instead of the file in STATA format, we also made available part of the `titanic` data set in tab-delimited format (`titanic3select.txt`). Download this file from the course website and then import it using the command `read.table`. Note that this file has been manipulated in Excel and that importing the data is not as straightforward as you would have hoped. You will need to tweak the arguments of `read.table` and/or make some changes to the file manually.

Answer

Running the following line of code:

```
titanic3.select <- read.table("titanic3select.txt",sep="\t",header=TRUE)
```

throws an error message:

```
Error in scan(file = file, what = what, sep = sep, quote = quote, dec = dec,  :
line 16 did not have 18 elements Error during wrapup: cannot open the connection
```

This error message can in principle be corrected by adding the argument `fill=TRUE`, which adds blank fields if rows have unequal length. However, running

```
titanic3.select <- read.table("titanic3select.txt",sep="\t",header=TRUE,fill=TRUE)
```

throws a warning message:

```
Warning message: In scan(file = file, what = what, sep = sep, quote = quote, dec =
dec,  :   EOF within quoted string
```

It turns out that the data was not correctly imported, which is often easily diagnosed by inspecting the dimensions:

```
dim(titanic3.select)
```

Indeed the number of rows is 25 instead of the expected 30. This is caused by a trailing ' at row 25 for variable `home_dest`. This can be corrected by changing the default set of quoting characters:

```
titanic3.select <- read.table("titanic3select.txt", sep="\t", header=TRUE, fill=TRUE,
quote="\"")
```

A closer inspection of the data shows that the last column only contains NAs caused by trailing spaces. On the course website there is a corrected version of the file where the last column and the trailing ' at row 25 were deleted.

```
titanic3.select <- read.table("titanic3select_corrected.txt",sep="\t",header=TRUE)
```

Question 10. We have a further look at the output from the `summary` function, which was not always what we would like to see. First, give the commands (`sapply` will be explained later):

```
sapply(titanic3, mode)
```

```
#      pclass    survived        name         sex         age       sibsp
#   "numeric"   "numeric" "character"   "numeric"   "numeric"   "numeric"
#       parch      ticket        fare       cabin    embarked        boat
#   "numeric" "character"   "numeric"   "numeric"   "numeric"   "numeric"
#        body   home.dest         dob      family      agecat
#   "numeric" "character"   "numeric"   "numeric"   "numeric"
```

```
sapply(titanic3, is.factor)
```

```
#    pclass  survived      name       sex       age     sibsp     parch    ticket
#      TRUE     FALSE     FALSE      TRUE     FALSE     FALSE     FALSE     FALSE
#      fare     cabin  embarked      boat      body home.dest       dob    family
#     FALSE      TRUE      TRUE      TRUE     FALSE     FALSE     FALSE      TRUE
#    agecat
#      TRUE
```

We see that `survived` has mode "numeric" and is not a `factor` (`is.factor(titanic3$survived)` gives `FALSE`). It is interpreted in the same way as the truly continuous variable `age`.

(a) Add a variable `status` to the `titanic` data set, which gives the survival status of the passengers as a factor variable with labels "no" and "yes" describing whether individuals survived. Make the value "no" the first level (see titanic3info.txt for the reason).

Answer

```
# titanic3info.txt: Survival (0 = No; 1 = Yes)
titanic3$status <- factor(titanic3$survived, labels=c("no","yes"))
```

(b) Variable `pclass` has mode "numeric" and is a `factor`, whereas `home.dest` has mode "character" and is not a `factor`. Give the commands

```
as.numeric(head(titanic3$pclass))
as.numeric(head(titanic3$home.dest))
```

and explain the difference in output.

Answer

```
as.numeric(head(titanic3$pclass))
```

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

```
as.numeric(head(titanic3$home.dest))

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

We do not change the variables in this dataset that have mode “character”. They are variables that have many different values, and there is no reason to convert them into factors.

Question 11. **(a)** Take a look at the name (`name`), and the home town/destination (`home.dest`) of all passengers who were older than 70 years. Use the appropriate selection functions.

Answer

```
# Note that the function 'subset' also leaves out the passsengers for which the
# variable 'age' is missing
subset(titanic3,age>70)[,c("name","home.dest")]

#                                  name               home.dest
# 10             Artagaveytia, Mr. Ramon     Montevideo, Uruguay
# 15     Barkworth, Mr. Algernon Henry W           Hessle, Yorks
# 62      Cavendish, Mrs. Tyrell William Little Onn Hall, Staffs
# 136          Goldschmidt, Mr. George B            New York, NY
# 728               Connors, Mr. Patrick
# 1236               Svensson, Mr. Johan
# Another solution is
subset(titanic3,age>70,select=c(name,home.dest))

#                                  name               home.dest
# 10             Artagaveytia, Mr. Ramon     Montevideo, Uruguay
# 15     Barkworth, Mr. Algernon Henry W           Hessle, Yorks
# 62      Cavendish, Mrs. Tyrell William Little Onn Hall, Staffs
# 136          Goldschmidt, Mr. George B            New York, NY
# 728               Connors, Mr. Patrick
# 1236               Svensson, Mr. Johan
# Yet another solution is
titanic3[titanic3$age>70 & !is.na(titanic3$age),c("name","home.dest")]

#                                  name               home.dest
# 10             Artagaveytia, Mr. Ramon     Montevideo, Uruguay
# 15     Barkworth, Mr. Algernon Henry W           Hessle, Yorks
# 62      Cavendish, Mrs. Tyrell William Little Onn Hall, Staffs
# 136          Goldschmidt, Mr. George B            New York, NY
# 728               Connors, Mr. Patrick
# 1236               Svensson, Mr. Johan
```

(b) There is one person from Uruguay in this group. Select the single record from that person. Did this person travel with relatives?

Answer

```
subset(titanic3, name=="Artagaveytia, Mr. Ramon")

#    pclass survived                    name  sex age sibsp parch   ticket
# 10    1st        0 Artagaveytia, Mr. Ramon male  71     0     0 PC 17609
#       fare cabin  embarked boat body           home.dest       dob family
# 10 49.5042       Cherbourg        22 Montevideo, Uruguay -43359.75     no
#     agecat status
# 10 (40,80]     no
# No he didn't travel with relatives, since the variable 'family' is 'no'
```

(c) Make a table of survivor status by sex, but only for the first class passengers. Use the `xtabs` function.

Answer

```
xtabs(~sex+status, data=titanic3, subset=(pclass=="1st"))

#         status
# sex       no yes
#   female   5 139
#   male   118  61
```

5 Graphics

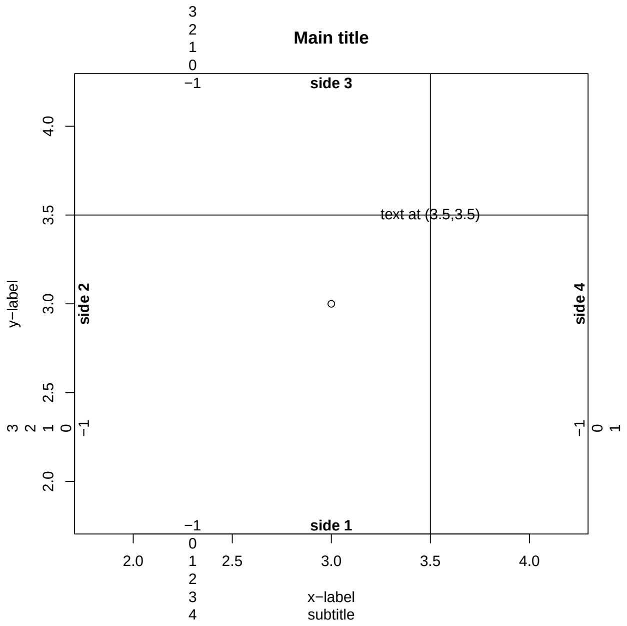
5.1 Basic graphics

The graphics subsystem of R offers a very flexible toolbox for high-quality graphing. There are typically three steps to producing useful graphics:

- Creating the basic plot
- Enhancing the plot with labels, legends, colors *etc.*
- Exporting the plot from R for use elsewhere

In the graphics model that R uses, a figure region consists of a central plotting region surrounded by margins. The basic graphics command is `plot`. The following piece of code illustrates the most common options:

```
plot(3,3,main="Main title",sub="subtitle",xlab="x-label",ylab="y-label")
text(3.5,3.5,"text at (3.5,3.5)")
abline(h=3.5,v=3.5)
for (side in 1:4) mtext(-1:4,side=side,at=2.3,line=-1:4)
mtext(paste("side",1:4),side=1:4,line=-1,font=2)
```



In this figure one can see that

- Sides are labelled clockwise starting with x-axis
- Coordinates in the margins are specified in *lines of text*
- Default margins are not all wide enough to hold all numbers -1,. . . ,4

You might want to use the `help` function to investigate some of the other functions and options.

Plots can be further finetuned with the `par` function. For instance, the default margin sizes can be changed using `par`. The default settings are

```
par("mar")
```

```
# [1] 5.1 4.1 4.1 2.1
```

This explains why only side 1 in the figure had a wide enough margin. This can be remedied by setting

```
par(mar=c(5,5,5,5))
```

before plotting the figure.

Question 12. (a) Try making this figure yourself by executing the code shown above.

(b) Save the figure you just made to a file. For this you have to know that R sends graphics to a *device*. The default device on most operating systems is the screen, for example the “Plots” window in RStudio. Saving a figure, therefore, requires changing R’s current device. See `help(device)` for the options. Save the figure you just made to a `png` and a `pdf` file. Don’t forget to close the device afterwards.

Answer

```
png(file="figure.png")
# Add code here to plot the figure (including par(mar=c(5,5,5,5)))
dev.off()
pdf(file="figure.pdf")
# Add code here to plot the figure (including par(mar=c(5,5,5,5)))
dev.off()
```

(c) When saving a figure to file, default values for the figure size are used. Save the figure to a `pdf` file with width and height of 21 inches.

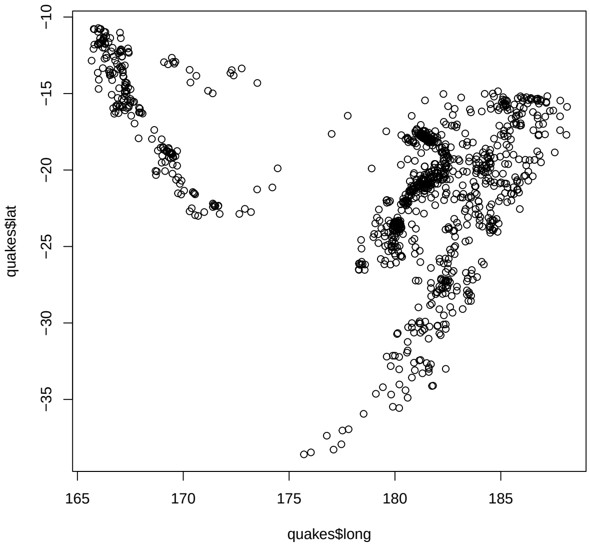
Answer

```
pdf(file="figureLarge.pdf",width=21,height=21)
# Add code here to plot the figure (including par(mar=c(5,5,5,5)))
dev.off()
```

Question 13. The `quakes` data set gives location and severity of earthquakes off Fuji. First load the data:

```
data(quakes)
```

(a) Make a scatterplot of latitude versus longitude. You should obtain a graph similar to:



Answer

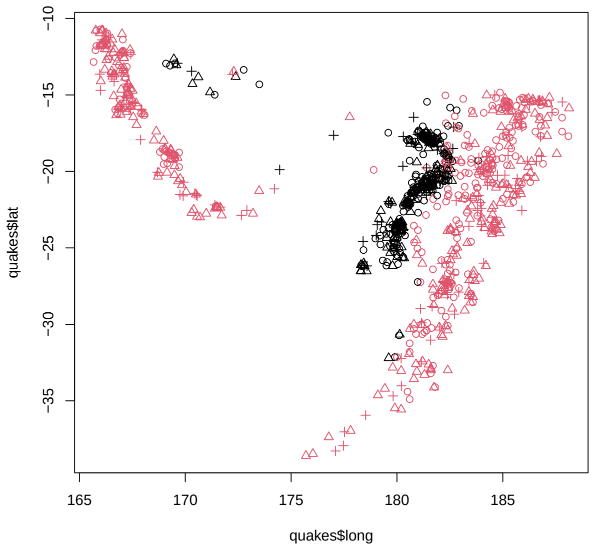
```
plot(quakes$long, quakes$lat)
```

(b) Use `cut` to divide the magnitude of quakes into three categories (cutoffs at 4.5 and 5) and use `ifelse` to divide depth into two categories (cutoff at 400). Hint: have a careful look at the (admittedly complicated) help page of `cut`, in particular the arguments `breaks` and `include.lowest`.

Answer

```
mag.cat <- with(quakes,cut(mag, breaks = c(4, 4.5, 5, 7),
                           include.lowest = TRUE ))
# More generic using the functions 'min' and 'max' that return the
# minimum and maximum, respectively, of a vector
mag.cat <- with(quakes,cut(mag, breaks = c(min(mag), 4.5, 5, max(mag)),
                           include.lowest = TRUE ))
levels(mag.cat) <- c("low", "medium", "high")
depth.cat <- factor(ifelse(quakes$depth>400, "deep", "shallow"))
```

(c) Redraw the plot, with symbols by magnitude and colors by depth. You should obtain a graph similar to:



Answer

```
# Note that for 'col' the factor 'depth.cat' is interpreted as numeric (deep=1 and
# shallow=2) and that the first two colours of palette() are used. Strangely enough
# for 'pch' this is not the case and the factor ''mag.cat' has to be converted to a
# numeric vector using the function 'as.numeric'.
plot(quakes$long, quakes$lat, pch=as.numeric(mag.cat), col=depth.cat)
```

(d) The magnitude of the earthquakes is given in Richter scale. Calculate the energy released in each earthquake as $10^{(3/2)}$ to the power of the Richter magnitude.

Answer

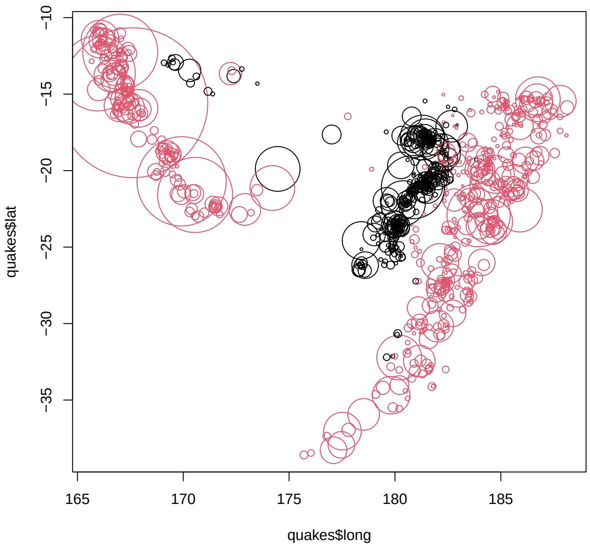
```
energy <- (10^(3/2))^quakes$mag
```

(e) The `cex` argument of `plot` can be used to scale the plot symbols. We will scale the plot symbols so that the surface of the plot symbol is proportional to the released energy. Calculate plot symbol size as the square root of the energy divided by the median square root of the energy (to get median symbol size 1).

Answer

```
symbolsize <- sqrt(energy)/median(sqrt(energy))
```

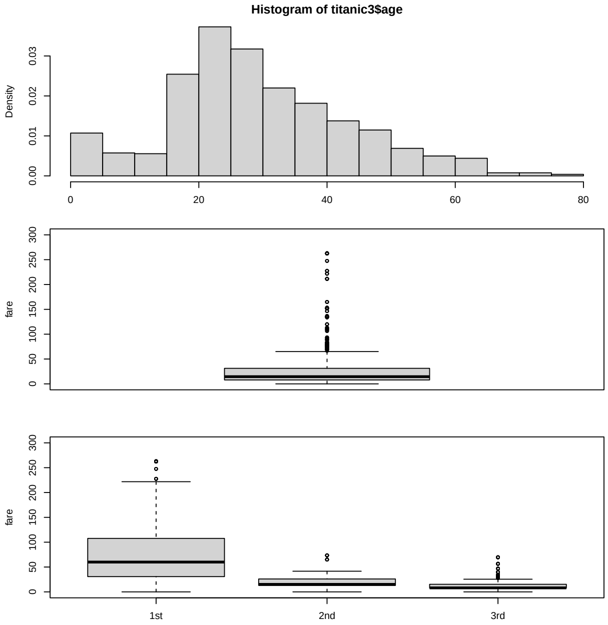
(f) Plot the magnitude of earthquakes again, but with plot symbols sized by energy. Keep the coloring by depth. You should obtain a graph similar to:



Answer

```
plot(quakes$long, quakes$lat, cex=symbolsize, col=depth.cat)
```

Question 14. Different plots can be combined in a single window (device) via , e.g., `par(mfrow=c(..))` or `layout(..)`. Combine the histogram and the two boxplots for the `titanic` data from the lecture into a single plot. Find a nice layout of your final plot, see help files for setting proper margins, etc. You should obtain a graph similar to:



Answer

```
par(mfrow=c(3,1))
# Make the margins smaller than default
par(mar=c(2,4,2,2))
hist(titanic3$age,breaks=15,freq=FALSE)
boxplot(titanic3$fare,ylim=c(0,300),ylab="fare")
plot(fare~pclass,data=titanic3,ylim=c(0,300),ylab="fare")
```

6 Structure of R

Question 15. Investigate which environments are in the search path. Take a look at the objects that exist in the workspace. Which is the current working directory?

Answer

```
# You will get different results depending on your configuration
search()
```

```
#  [1] ".GlobalEnv"        "package:foreign"   "package:stats"
#  [4] "package:graphics"  "package:grDevices" "package:utils"
#  [7] "package:datasets"  "package:methods"   "Autoloads"
# [10] "package:base"
```

```
ls()
```

```
#  [1] "curr"                        "denominator"
#  [3] "depth.cat"                   "energy"
#  [5] "exrate"                      "index"
#  [7] "islands"                     "islands.more20"
#  [9] "islands.names"               "islands.withoutMoluccas"
# [11] "M"                           "mag.cat"
# [13] "mtcars"                      "notMoluccas"
# [15] "numerator"                   "quakes"
# [17] "side"                        "symbolsize"
# [19] "titanic3"                    "vec"
# [21] "vowels"                      "xmean"
# [23] "xstd"
```

```
getwd()
```

```
# [1] "C:/Users/pdmoerland/Dropbox/Education/Computing in R/Exercises"
```

Question 16. The function `mean` is defined in the **base** package, which is included in the search path at startup. From the `search` command, we can see where in the search path the **base** package is located. Issue the command

```
ls("package:base", pattern="mean")
```

Instead of the argument `package:base`, we could have given the position in the search path of the **base** package (that is `ls(10, pa="mean")` if base is in position 10). Have a look at the different names of the `mean` function.

Question 17. Save the `titanic` data set in R binary format with extension ".RData".

Answer

```
save(titanic3, file="Titanic.RData")
```

7 Data manipulation

Question 18. Sort the `titanic` data set according to the age of the passengers and store the result in a separate object, e.g. named `data.sorted`. Have a look at the 10 youngest and the 10 oldest individuals. For reasons of space, restrict to the first 5 columns when showing the results. What do you notice with respect to passenger class and age?

Answer

```
# If not installed yet, first install the package dplyr that contains the function
# arrange
#install.packages("dplyr")
library(dplyr)
data.sorted <- arrange(titanic3, age)
head(data.sorted[,1:5],10)
```

```
#      pclass survived                              name    sex    age
# 764     3rd        1 Dean, Miss. Elizabeth Gladys \\"M female 0.1667
# 748     3rd        0   Danbom, Master. Gilbert Sigvard   male 0.3333
# 1241    3rd        1   Thomas, Master. Assad Alexander   male 0.4167
# 428     2nd        1         Hamalainen, Master. Viljo   male 0.6667
# 658     3rd        1            Baclini, Miss. Eugenie female 0.7500
# 659     3rd        1     Baclini, Miss. Helene Barbara female 0.7500
# 1112    3rd        0    Peacock, Master. Alfred Edward   male 0.7500
# 360     2nd        1     Caldwell, Master. Alden Gates   male 0.8333
# 549     2nd        1   Richards, Master. George Sibley   male 0.8333
# 612     3rd        1         Aks, Master. Philip Frank   male 0.8333
```

```
# Use is.na to exclude the passengers for whom 'age' is missing
tail(subset(data.sorted,!is.na(age))[,1:5],10)
```

```
#      pclass survived                            name    sex  age
# 595     2nd        0           Wheadon, Mr. Edward H   male 66.0
# 286     1st        0              Straus, Mr. Isidor   male 67.0
# 82      1st        0    Crosby, Capt. Edward Gifford   male 70.0
# 507     2nd        0     Mitchell, Mr. Henry Michael   male 70.0
# 728     3rd        0            Connors, Mr. Patrick   male 70.5
# 10      1st        0         Artagaveytia, Mr. Ramon   male 71.0
# 136     1st        0       Goldschmidt, Mr. George B   male 71.0
# 1236    3rd        0             Svensson, Mr. Johan   male 74.0
# 62      1st        1  Cavendish, Mrs. Tyrell William female 76.0
# 15      1st        1 Barkworth, Mr. Algernon Henry W   male 80.0
```

Question 19. Give a summary of the fare paid for the three passenger classes separately, using the `summary` function, the `subset` function for selecting the appropriate rows, as well as one of the mechanisms for selecting columns.

Answer

```
summary(subset(titanic3, pclass=="1st")$fare)
```

```
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#    0.00   30.70   60.00   87.51  107.66  512.33
```

```
summary(subset(titanic3, pclass=="2nd")$fare)
```

```
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#    0.00   13.00   15.05   21.18   26.00   73.50
```

```
summary(subset(titanic3, pclass=="3rd")$fare)
```

```
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#    0.00    7.75    8.05   13.30   15.25   69.55       1
```

Remark

Instead of writing three lines of code, applying the `summary` function to each of the three subsets, we could write a `for` loop. When using a `for` loop, we need to explicitly write the `print` command:

```
for(i in 1:3){print(summary(subset(titanic3, pclass==levels(pclass)[i])$fare))}
```

The function `aggregate` is yet another option:

```
aggregate(fare~pclass,data=titanic3,FUN=summary)
```

Later, we will see even more efficient ways.

Question 20. Create an extra variable that categorizes the variable `sibsp` (the number of siblings/spouses aboard) into three groups: 0, 1 and more than 1. Also, create an extra factor variable named `paid`, which shows whether the individual paid for the trip (i.e. whether fare >0). Preferably, use the `within` or the `transform` function. Check whether the results are correct.

Answer

```
titanic3 <- within(titanic3, {
   sibspcat <- cut(sibsp, breaks=c(0,1,2,9),include.lowest=TRUE,
                      right=FALSE, labels=c("0","1","2-8"))
   paid <- factor(fare>0, labels=c("no","yes"))
} )

titanic3 <- transform(titanic3,
   sibspcat = cut(sibsp, breaks=c(0,1,2,9),include.lowest=TRUE,
     right=FALSE, labels=c("0","1","2-8")),
   paid = factor(fare>0, labels=c("no","yes"))
 )
```

8 Documentation and help

Question 21. R is also very flexible with respect to manipulation of character data, such as words. In this exercise, you will see an example.

(a) Create a character vector of length three that consists of the words “Academic”, “Medical” and “Center”. Give the object the name `AMC`. Check the *mode* of the object.

Answer

```
AMC <- c("Academic","Medical","Center") # a character vector named AMC
mode(AMC)
```

```
# [1] "character"
```

```
is.character(AMC)
```

```
# [1] TRUE
```

(b) Next, we want to abbreviate the word and obtain `AMC` as output. Try to find the appropriate functions using the commands

```
help.search("abbreviate")
help.search("combine")
help.search("quote")
```

Answer

```
#help(abbreviate)
abbreviate(AMC,1) # selects first character
```

```
# Academic  Medical   Center
#      "A"      "M"      "C"
```

```
#help(paste)
paste(abbreviate(AMC,1),collapse="") # gives AMC
```

```
# [1] "AMC"
```

Finally, if you want to remove the quotes give the following command

```
noquote(paste(abbreviate(AMC,1),collapse="")) # removes quotes
```

Question 22. Try to find more information on a topic of interest and how R can be of help. If you do not have any idea, you can search for the keyword “crosstab”.

Answer

```
help(crosstab)
help.search("crosstab")
RSiteSearch("crosstab")
install.packages("sos")
library(sos)
findFn("crosstab")
```

9 Statistical analysis

Question 23. Fit a linear model that predicts fare as a function of age. Since fare has a very skewed distribution, we use the transformed variable `log10(fare+1)`. Consider the following issues.

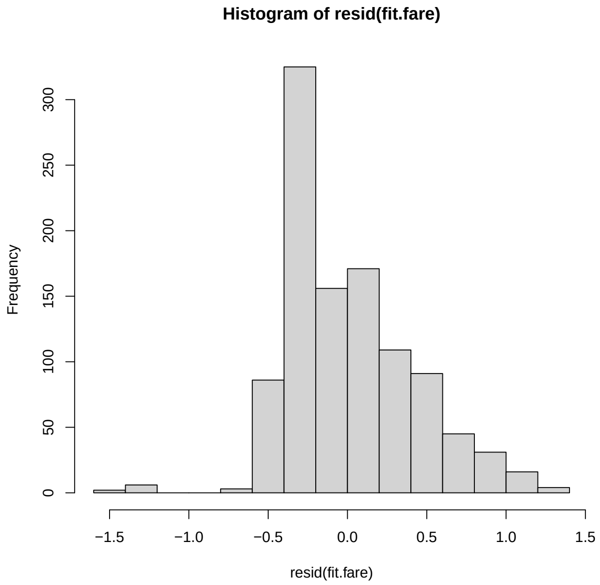
(a) Fit the model and store the results in an R object. Summarize the model using the `summary` function.

Answer

```
fit.fare <- lm(log10(fare+1) ~ age, data=titanic3)
summary(fit.fare)
```

```
#
# Call:
# lm(formula = log10(fare + 1) ~ age, data = titanic3)
#
# Residuals:
#      Min       1Q   Median       3Q      Max
# -1.44366 -0.33354 -0.07742  0.27298  1.34630
#
# Coefficients:
#              Estimate Std. Error t value Pr(>|t|)
# (Intercept) 1.1651838  0.0293881  39.648  < 2e-16 ***
# age         0.0056833  0.0008869   6.408 2.23e-10 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 0.4123 on 1043 degrees of freedom
#   (264 observations deleted due to missingness)
# Multiple R-squared:  0.03788, Adjusted R-squared:  0.03696
# F-statistic: 41.06 on 1 and 1043 DF,  p-value: 2.228e-10
```

(b) One of the assumptions of a standard linear model is that the residuals have approximately normal distribution. Make a histogram of the residuals, using the functions `resid` and `hist`. You should obtain a graph similar to:



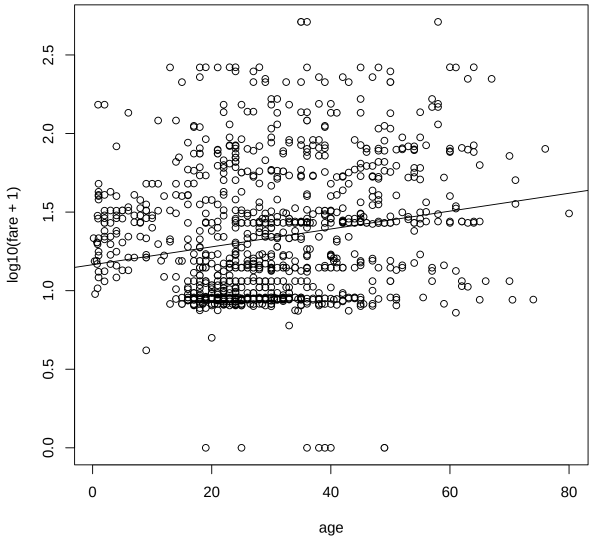
Answer

```
fit.fare <- lm(log10(fare+1) ~ age, data=titanic3)
hist(resid(fit.fare))
```

(c) Make a plot of the residuals against the fitted values, using (with `fit.fare` the name of the object that contains the linear model fit):

```
plot(resid(fit.fare)~fitted(fit.fare))
```

(d) Make a scatterplot of fare against age and add the linear regression line. A fitted regression line is added to a plot via `abline(fit.fare)`. You should obtain a graph similar to:



Answer

```
plot(log10(fare+1)~age,data=titanic3)
abline(fit.fare)
```

(e) Does the object have a class? If so, which are the generic functions that have a method for this class?

Answer

```
class(fit.fare)
```

```
# [1] "lm"
```

```
methods(class="lm")
```

```
#  [1] add1           alias          anova          case.names     coerce
#  [6] confint        cooks.distance deviance       dfbeta         dfbetas
# [11] drop1          dummy.coef     effects        extractAIC     family
# [16] formula        hatvalues      influence      initialize     kappa
# [21] labels         logLik         model.frame    model.matrix   nobs
# [26] plot           predict        print          proj           qr
# [31] residuals      rstandard      rstudent       show           simulate
# [36] slotsFromS3    summary        variable.names vcov
# see '?methods' for accessing help and source code
```

10 Programming and ply functions

Functions from the `apply` family are convenient shorthands for repetitions.

Question 24. Use `apply` to calculate the mean of the variables `age`, `fare`, and `body` of `titanic3`.

Answer

```
apply(subset(titanic3,select=c("age","fare","body")),2,mean,na.rm=TRUE)
```

```
#       age      fare      body
#  29.88113  33.29548 160.80992
```

Question 25. The `chickwts` data describes chicken weights by feed type. First load the data:

```
data(chickwts)
```

(a) Calculate the mean weight for each feed type.

Answer

```
tapply(chickwts$weight, chickwts$feed, mean)
```

```
#    casein horsebean   linseed  meatmeal   soybean sunflower
#  323.5833  160.2000  218.7500  276.9091  246.4286  328.9167
```

(b) Count the number of chicks with weight over 300 grams,

Answer

```
sum(chickwts$weight > 300)
```

```
# [1] 26
```

Further abstraction of the R code is possible through *functions*. Functions lead to more compact code that is easier to understand and also avoid duplication of the same code over and over again.

```
name <- function(arg_1,arg_2, ...){
   expr
   }
```

- `expr` is, in general, a grouped expression containing the arguments `arg_1, arg_2` ...
- A call to the function is of the form `name(expr_1,expr_2, ...)`
- The value of the expression `expr` is the value returned by the function

Question 26. **(a)** For the `chickwts` data, write a function that takes a vector `x` as input and returns the number of observations in `x` greater than 300.

Answer

```
greater300 <- function(x) {
  sum(x > 300)
}
```

(b) Calculate the number of chicks with weight over 300 grams for each feed type.

Answer

```
tapply(chickwts$weight, chickwts$feed, greater300)
```

```
#    casein horsebean   linseed  meatmeal   soybean sunflower
#         8         0         1         5         3         9
```