R Programming: Worksheet 2

After today you should know:

- the differences between a data frame and a matrix;
- how to access parts of a data frame using `subset()`;
- how to alter the elements of a data frame, including its row names and variable names;
- what a factor is, and how to create one from a character vector;
- how to get boxplots using factors.

1. **Data Frames**

   Load the `MASS` library and take a look at the dataset called `survey`.

   ```
   > library(MASS)
   > head(survey)
   ```

   You can look at the documentation:

   ```
   > ?survey
   ```

   and get a brief summary of each variable:

   ```
   > summary(survey)
   ```

   (a) Find the mean pulse rate of the students. What goes wrong here? *There are missing values, so*

   ```
   > mean(survey$Pulse)
   ```

   leads to `NA`.

   The vector of pulses stored in `survey` contains some entries which are labelled `NA`. This is used in `R` to represent **missing data**.

   (b) Try looking at the documentation for `mean()` to see how to get around this. *Inspection of the documentation shows that you can get around this by using*

   ```
   > mean(survey$Pulse, na.rm = TRUE)

   ## [1] 74.15
   ```

   which ignores missing values for the purposes of calculating the mean.

   (c) The ages are recorded as fractions representing a number of months. Change that columns of the data frame so that it contains whole years (the `floor()` command may be useful).

```
> survey$Age = floor(survey$Age)
```

Another possibility is:

```
> survey = within(survey, Age <- floor(Age))
```

(d) Find the mean pulse rate for students under 20.

```
> with(survey, mean(Pulse[Age < 20], na.rm = TRUE))

## [1] 75.49
```

Subsetting. Suppose I want to obtain the records of students who are over 190 cm tall. Since data frames allow me to subset just like a matrix, I might just think of typing:

```
> survey[survey$Height > 190, ]
```

What goes wrong here? *Every missing height value leads to an `NA` in the logical vector we're subsetting with. This leads (in each case) to a row of `NA`s in the data frame.* Try instead the following:

```
> subset(survey, Height > 190)
```

The subset function ignores missing values, which is usually the behaviour we would prefer. We can also select only some of the fields of the data frame if we prefer:

```
> subset(survey, Height > 190, select = c("Pulse", "Clap"))
```

Recall that the `&` operator does a point-wise logical 'and' comparison.

```
> subset(survey, (Pulse > 70) & (Smoke == "Heavy"))
```

Similarly, `|` is for 'or', and `!` for 'not'.

```
> with(survey, (Pulse > 70) | (Pulse < 45))
> !(survey$Age > 30)
```

(e) Find the mean age of students who write with their right hand. *Either of:*

```
> mean(subset(survey, W.Hnd == "Right")$Age)
> mean(survey$Age[survey$W.Hnd == "Right"], na.rm = TRUE)
```

In the second case the `na.rm=TRUE` is necessary, even though no ages are missing.

(f) What proportion of left handers do not clap with their left hand on top?

```
> mean(subset(survey, W.Hnd == "Left", select = Clap) != "Left")

## [1] 0.5

> mean(survey$Clap[survey$W.Hnd == "Left"] != "Left", na.rm = TRUE)

## [1] 0.5
```

(g) Using the `plot()` command, plot the pulse of the subjects against their age.

```
> plot(survey$Age, survey$Pulse)
```

Try subtracting 10 from the age and taking the logarithm (using the function `log()`), to obtain a slightly clearer picture.

```
> with(survey, plot(log(Age - 10), Pulse))
```

2. **More Manipulation**

Load the `hills` dataset from lectures. What happens when you apply the `plot()` command to this data? *It produces pair-wise scatter plots of the three variables.*

```
> plot(hills)
```

You can access the row names of the `hills` data with the function `rownames()`. You can also **change** the rownames this way:

```
> rownames(hills)[1] <- "Redmantle"
> rownames(hills)[1]
```

If at any point you want to revert to the original data frame, type

```
> rm(hills)
```

You can't really edit the original data set in the `MASS` package, only your own copy. Calling `rm()` deletes the copy.

(a) The race climbs are measured in feet. Change the data set so that they are measured in metres (there are 2.54 cm in 1 inch, and 12 inches in 1 foot). Use the `round()` command to set them to the nearest 10 metres.

```
> x = 100/(12 * 2.54)   # feet in a meter
> x

## [1] 3.281

> hills$climb = round(hills$climb/x, -1)
```

(b) Obtain a logical vector indicating which races were longer than an hour. How many are this long?

```
> hills$time > 60

##  [1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE
## [12] FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
## [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE
## [34] FALSE  TRUE

> sum(hills$time > 60)

## [1] 11
```

The `which()` command is a useful way to turn a logical vector into a numerical one. For example, try

```
> which(hills$dist < 5)

##  [1]  1 15 18 19 21 22 24 25 26 34
```

(c) Use `which()` to select the first three races under 5 miles.

```
> hills[which(hills$dist < 5)[1:3], ]

##             dist climb  time
## Greenmantle  2.5   200 16.08
## Eildon Two   4.5   460 26.93
## Knock Hill   3.0   110 78.65
```

(d) Identify (with an `R` command, not just by looking) which of the races involves `"Meall Ant-Suidhe"`. I've no idea how to pronounce Meall Ant-Suidhe, so remove it from the data frame (this might not normally be considered good statistical practice).

```
> rm = which(rownames(hills) == "Meall Ant-Suidhe")
> hills = hills[-rm, ]
```

(e) Plot the race times against distance. Can you spot an outlier? (think carefully)

```
> plot(hills$dist, hills$time)
```

One of the very short races (5 miles) has a record time of over 78 minutes; closer inspection reveals no significant climbs.

(f) In fact there is a race whose time was recorded as one hour larger than it ought to have been. Correct this.

```
> which(hills$dist < 4 & hills$time > 50)

## [1] 18

> hills[18, ]
```

```
##              dist climb  time
## Knock Hill    3   110 78.65

> hills[18, 3] = hills[18, 3] - 60
```

You can turn the data frame into a matrix like this:

```
> hillsMat = as.matrix(hills)
```

(g) Print the two objects `hills` and `hillsMat` by typing their names. Can you see any difference? *Superficially they look the same. `print.default()` tells a different story.*

(h) Use `is()` to see that they *are* different. What happens if you use `plot()` on the matrix? Compare this behaviour with that of the command `pairs()`. *`plot()` when applied to a numeric matrix just uses the first two columns. `pairs()` does the right thing regardless.*

(i) What happens if you try to turn the `survey` data into a matrix? *Everything in a matrix has to be of the same type, so it makes everything into a string (character vector).*

3. **Factors**

(Extension of Exercise 3.5 from Lectures)

Take a look at the `birthwt` data from the `MASS` package.

(a) How is race stored in these data? Is this sensible? *It's stored as a numeric vector, which isn't sensible since these data are categorical.*

(b) Turn this into a factor with level names as indicated in the documentation. *One way is*

```
> birthwt$race <- factor(birthwt$race, labels = c("white", "black", "other"))
```

but you might also turn it into a character vector first:

```
> birthwt$race <- as.factor(c("white", "black", "other")[birthwt$race])
```

(c) Use the `table()` command to count the number of babies of each race.

```
> table(birthwt$race)

##
## white black other
##    96    26    67
```

(d) Try the following command:

```
> tab = with(birthwt, table(smoke, low))
```

What do the results in `tab` suggest? *Comparing*

```
> tab[1, 2]/sum(tab[1, ])

## [1] 0.2522

> tab[2, 2]/sum(tab[2, ])

## [1] 0.4054
```

it seems that a much higher proportion of babies whose mother smoked during pregnancy were of low birthweight (although the documentation doesn't specify how the data were collected, so we have to be careful what we conclude from this.)

4. **Reading In Data**

   I have emailed you a dataset called `sprays.dat`. The data represent insect counts in agricultural experiments treated with different insecticides. Save the file onto your local drive.

   (a) Read the data into `R` using the command

   ```
   > dat = read.table("sprays.dat", header = TRUE)
   ```

   Check that the first row is as you expect.

   ```
   > head(dat)

   ##   count spray
   ## 1    10     A
   ## 2     7     A
   ## 3    20     A
   ## 4    14     A
   ## 5    14     A
   ## 6    12     A
   ```

   What happens if you omit the `header=TRUE` part? *The first row ends up being the variable names.*

   (b) Look at the two variables in this new data frame; what types do they have?difference between them do you notice? *The variable `spray` has automatically become a factor, but `count` is a numeric vector.* Is their format appropriate? *Yes, the factor is sensible for categorical data.*

   (c) Find the mean number of insects for each different experimental unit.

   ```
   > mean(dat$count[dat$spray == "A"])

   ## [1] 14.5

   > mean(dat$count[dat$spray == "B"])

   ## [1] 15.33

   > # ETC
   ```

(d) Look at the documentation for the command `tapply()`. Now repeat the previous question with a single command.

```
> tapply(dat$count, dat$spray, mean)

##      A      B      C      D      E      F
## 14.500 15.333  2.083  4.917  3.500 16.667
```

(e) Try using the command `quantile()` on the counts. Use `tapply()` to find the upper and lower quartiles of the data broken down by spray type.

```
> quantile(dat$count)

##    0%   25%   50%   75%  100%
##  0.00  3.00  7.00 14.25 26.00

> tapply(dat$count, dat$spray, quantile)

## $A
##    0%   25%   50%   75%  100%
##  7.00 11.50 14.00 17.75 23.00
##
## $B
##   0%  25%  50%  75% 100%
##  7.0 12.5 16.5 17.5 21.0
##
## $C
##   0%  25%  50%  75% 100%
##  0.0  1.0  1.5  3.0  7.0
##
## $D
##    0%   25%   50%   75%  100%
##  2.00  3.75  5.00  5.00 12.00
##
## $E
##   0%  25%  50%  75% 100%
## 1.00 2.75 3.00 5.00 6.00
##
## $F
##   0%  25%  50%  75% 100%
##  9.0 12.5 15.0 22.5 26.0
```

(f) * Can you write a command which gives you just a vector of the six upper quartiles? [Hint: how can you pass the argument `probs=0.75` to `quantile()` when you are using `tapply()`?]

```
> tapply(dat$count, dat$spray, quantile, 0.75)

##     A     B     C     D     E     F
## 17.75 17.50  3.00  5.00  5.00 22.50
```

(g) Use the `plot()` command to produce separate boxplots for each level (as we did in class for the height data).

```
> plot(dat$spray, dat$count)
```