MATH10282 Introduction to Statistics
Semester 2, 2020/2021
Examples Class 3: R Session

Introduction

In this second R session we will see how we can generate commonly used sequences of numbers, calculate various summary statistics and also how to create some simple graphical displays. All the data sets required are in the course website `https://minerva.it.manchester.ac.uk/~saralees/intro.html`

The functions `rep` and `seq`

The function `rep`

This is used to replicate values in a vector `x`, which is one of the arguments of the function. Its usage is:

```
rep(x, ...)
```

where `x` is a vector of values to be replicated and `...` denotes further optional arguments which may be used. These include:

- `times` - a vector giving the number of times to repeat elements of `x`. If `times` is of length `length(x)`, then ith element of `x` is repeated `times[i]` times. If `times` is of length 1, the whole vector is repeated. The default is `times=1`.
- `length.out` - this is a non-negative integer which gives the desired length of the output vector.
- `each` - is a non-negative integer. Each element of `x` is repeated `each` times. The default value is 1.

Some examples of its use are:

```
> x<-c(3, 22, 5, 7)
> rep(x, times=2)
[1]  3 22  5  7  3 22  5  7
> rep(x, times=c(1, 2, 2 ,3))
[1]  3 22 22  5  5  7  7  7
> rep(x, length.out=8)
[1]  3 22  5  7  3 22  5  7
> rep(x, each=3)
 [1]  3  3  3 22 22 22  5  5  5  7  7  7
>
>
> rep(1:3, times=2)
[1] 1 2 3 1 2 3
> rep(1:3, each=2)
[1] 1 1 2 2 3 3
> rep(1:3, times=c(1:3))
[1] 1 2 2 3 3 3
>
>
> rep(c("A", "B"), times=c(2,3))
[1] "A" "A" "B" "B" "B"
```

The function `seq`

This is used to generate regular sequences of numbers. Its form of use is `seq(...)`, where `...` denotes optional arguments which may be used. These include:

- `from, to` - these are the start and end values of the sequence.
- `by` - a number which gives the increment of the sequence.
- `length.out` - a non-negative integer giving the desired length of the sequence.
- `along.with` - take the length of the sequence from the length of this argument.

Some examples of its use are:

```
> seq(from=2, to=8, by=2)
[1] 2 4 6 8
> seq(from=2, to=8, by=3)
[1] 2 5 8
> seq(from=2, to=8, by=3.5)
[1] 2.0 5.5
> seq(from=5, to=2, by=-0.5)
[1] 5.0 4.5 4.0 3.5 3.0 2.5 2.0
> seq(from=2, to=8, length.out=6)
[1] 2.0 3.2 4.4 5.6 6.8 8.0
>
>
> z=c(1:5)
> seq(from=4, to=14, along.with=z)
[1]  4.0  6.5  9.0 11.5 14.0
>
>
> 1:4 # note that for successive integers we
           do not need to use seq
[1] 1 2 3 4
> 4:1
[1] 4 3 2 1
```

Simple summary statistics

(i) **The mean and median**. If we have data in a vector `x` then the commands `mean(x)` and `median(x)` calculate the mean and median, of the data in `x`, respectively.

(ii) **The variance and standard deviation**. For data in a vector `x`, we can calculate the variance and standard deviation using the functions `var(x)` and `sd(x)`, respectively.

(iii) **Five number summary**. We can use the quantile function as described in the lectures, i.e. `quantile(x, probs=c(0, 0.25, 0.5, 0.75, 1), type=6)` will produce estimates based on using $r = p \times (n + 1)$.

However, using `type=7` in this function is the default R method for this and other quantile estimation functions. This corresponds to using $r = p \times (n - 1) + 1$.

Use of the function `summary(x)` calculates the five number summary and also the sample mean. The results are printed out with labels indicating what each value is. The quantiles are estimated here using the method `type=7`.

Another alternative is to use the function `fivenum(x)` which results in a vector of length 5 containing the sample mininimum, $\hat{Q}(0.25)$, $\hat{Q}(0.5)$, $\hat{Q}(0.75)$ and sample maximum, in that order. Here the quantiles are estimated using another different methodology.

(iii) We can illustrate these functions on the Moses data.

```
> moses<-read.table(file="https://minerva.it.manchester.ac.uk/~saralees/moses.txt",
                             header=TRUE)
> names(moses)
[1] "time"
> t<-moses$time # copy the data into vector t for convenience
> mean(t)
[1] 48.52844
> median(t)
[1] 48.54
> var(t)
[1] 0.5354976
> sd(t)
[1] 0.731777
> fns<-fivenum(t) # store the five number summary in the vector fns
> fns
[1] 47.02 47.95 48.54 49.00 50.19
> iqr<-fns[4]-fns[2] # this calculates the iqr for the moses data
> iqr
[1] 1.05
>
> summary(t)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  47.02   47.96   48.54   48.53   48.99   50.19
>
> moses_q7=quantile(moses$time, probs=c(0, 0.25, 0.5,
              0.75, 1), type=7)
> moses_q7
     0%     25%     50%     75%    100%
47.0200 47.9575 48.5400 48.9950 50.1900
>
>
> moses_q6=quantile(moses$time, probs=c(0, 0.25, 0.5,
             0.75, 1), type=6)
> moses_q6
     0%     25%     50%     75%    100%
47.0200 47.9475 48.5400 49.0000 50.1900
```

Graphical displays of data

By default, R has just one graphics window open, called device 2, when producing a plot. When a new plot is produced later in the session the existing active device 2 is closed down and then reopened containing the new plot. The previous plot is not saved but the command(s) that produced it may be recovered by using the up and down arrow keys to scroll through your command history. You can save any plot by firstly clicking on the graphics window, using the menu command File→Save As, and then choosing the file format you want. You will then be asked to choose a filename and folder location to save to. Such plots can later be copied and pasted into a Word document if required.

On Windows, if you wish to record all the plots you produce then use:

```
> windows(record=T)
```

By default they will be saved in device 2 and can be scrolled through using the Page Up and Page Down keys. Closing the active graphics window will lose all your plots though. On Macintosh, by default the last ten plots are saved in the default device. These can be scrolled through using Cmd+←, Cmd+→.

You can open a new active device as follows. The new graphics device number will be one more than the largest existing device number.

```
> dev.new()
```

You can choose which graphics window to view by selecting it from the Windows drop down menu at the top of the R GUI. The command

```
> dev.set(3)
```

would make device 3 the active device, provided it already exists and then you can choose to view it from the Windows menu. The command

```
> dev.off(3)
```

closes down the specified device number. The command `graphics.off()` closes down all current graphics devices.

This may all sound a little complicated at first. My advice is to just use the default device 2 where possible. Plots may be permanently saved as described above, or be recovered by editing the command history. Recording your plotting history using `windows(record=T)` can be very useful.

The bar chart for discrete data

In an opinion poll from May 2015, the following preferences were counted: Conservatives 369, Labour 314, Liberal Democrat 75, UKIP 118, Other 124. We firstly create the data then tabulate the results and produce a bar chart as follows:

```
> poll <-rep(c("Conservative", "Labour", "Lib Dem", "UKIP", "Other"),
             times=c(369, 314, 75, 118, 124))
> poll <-factor(poll)
> table(poll)
poll
Conservative       Labour      Lib Dem        Other         UKIP
         369          314           75          124          118
> plot(poll)
```

You can add your own title using the `main="..."` optional argument in the plot command.

The histogram for continuous data

For data in a vector `x` the simplest way to create a histogram is by using the command `hist(x)`. For the Moses data the following command produces a *frequency* histogram, with R automatically choosing the number of bins (here 7), the bin width (here 2.5), and the origin:

```
> t<-moses$time
> hist(t)
```

For a *density* histogram we use the option `freq=F`. By default, `freq=T`, which produces a frequency histogram.[1]

If we also want to change the number of bins we use the option `breaks` as follows:

```
> hist(t, freq=F, breaks=14)
```

which doubles the number of bins to 14. You should notice that the resulting histogram is now much noisier than before. Try reducing the number of bins to 4 and note the form of the resulting histogram. Try other numbers of bins and note the differences between the histograms. Which number of bins do you think works best?

Now try completely specifying the bins using the `breaks` option to specify the mesh, e.g.

```
> hist(t, freq=F, breaks=seq(from=47.0, to=51.0, by=0.4))
> hist(t, freq=F, breaks=seq(from=46.8, to=50.8, by=0.4))
```

For these two histograms the bin width of 0.4 is the same but the origin differs by 0.2. Note the differences in the two plots.

In practice, it is usually best to start with the default bins option (i.e. do not use `breaks`) and then refine the number and/or their location if appropriate to obtain a 'better' histogram. Some degree of exploring different numbers of bins is generally a good idea.

You can add a title using the `main="..."` option.

The box plot for continuous data

To produce a box plot for the data in a vector `x` use the command `boxplot(x)`.

Suppose that `x` contains observations from two (or more) different groups. If we have a factor vector `grp` listing the group to which each observation belongs, then we can produce side by side box plots for the different groups using the command `boxplot(x~grp, data=" ")`, where `data=` specifies the data frame from which the variables `x` and `grp` should be taken.

For example,

```
> t<-moses$time
> boxplot(t)
```

produces a boxplot for the Moses data. You can change the label on the y-axis using the command `ylab="..."`

```
> runners<-read.table(file="https://minerva.it.manchester.ac.uk/~saralees/runners_group.txt",
                       header=T)
> names(runners)
[1] "pmol"  "group"
> runners
    pmol group
1   29.6     1
```

[1]Recall from Chapter 2 that a density histogram based on bins $B_1, \dots B_K$ of width h has $\text{Hist}(x) = \frac{v_k}{nh}$ for $x \in B_k$, where v_k is the number of observations in B_k. A density histogram integrates to 1 (Proposition 2.1). A frequency histogram instead has $\text{Hist}^{(f)}(x) = \frac{v_k}{h}$, and so it integrates to n.

```
2   25.1     1
3   15.5     1
4   29.6     1
5   24.1     1
6   37.8     1
7   20.2     1
8   21.9     1
9   14.2     1
10  34.6     1
11  46.2     1
12  66.0     2
13  72.0     2
14  79.0     2
15  84.0     2
16 102.0     2
17 110.0     2
18 123.0     2
19 144.0     2
20 162.0     2
21 169.0     2
22 414.0     2
> boxplot(pmol~group, data=runners)
```

Use `help(boxplot)` to explore the many other optional arguments.

Note that it is possible to extract interesting subsets of rows from a data set using logical vectors. For example, we can use the following code to extract the rows corresponding to runners with endorphin concentration greater than 100 pmol/l:

```
> highconc <- ( runners$pmol > 100 )
> runnersHi <- subset(runners, subset = highconc)
> runnersHi
   pmol     group
16  102 Collapsed
17  110 Collapsed
18  123 Collapsed
19  144 Collapsed
20  162 Collapsed
21  169 Collapsed
22  414 Collapsed
```

Note that after subsetting some factors may have unused levels, e.g. in the subset here, the level `Successful` is not used.

```
> summary(runnersHi$group)
 Collapsed Successful
         7          0
```

Sometimes it is helpful for further analysis to remove unused levels from consideration, as follows:

```
> runnersHi <- droplevels(runnersHi)
> summary(runnersHi$group)
Collapsed
        7
```

Scatter plots and curves

The `plot` command is very flexible and useful. One of its simplest uses is for creating scatter plots:

```
> x <- 0:5
> plot(x=x, y=x^2)
```

Another is for drawing curves specifed by pairs of (x, y) co-ordinates, e.g.:

```
> x <- seq(from=0, to=5, length=100)
> plot(x=x, y=x^2, type="l", col=2)
```

The `col` argument adds colour. Other curves can be added to the same axes as follows:

```
> lines(x=x,y=x^3, col=4)
```

Saving your R session

You can save the contents of your current workspace, i.e. all active objects but not plots, by using the menu item 'File'→'Save Workspace'. You will then be asked to choose a filename. Add the extension `.RData` to the filename you specify. Your command history can also be saved using 'File'→'Save History' and then specifying a filename with the `.Rhistory` extension.

At a later date you can resume this saved session by opening R and choosing 'File'→'Load Workspace' and then 'File'→'Load History'. In both cases you select the appropriate previously saved files.

Exercises

(i) The data are in the file `https://minerva.it.manchester.ac.uk/~saralees/simdat1.txt` which comprises two columns: the first contains case numbers and the second the data. The first row of the file contains the variable name for each column. These data are $n = 100$ observations simulated from a particular parametric distribution.

Explore these data in R using a range of summary statistics, as well as a histogram and box plot. Suggest a suitable parametric family from which the data could have arisen. What might be intuitively sensible estimates of any unknown parameters?

(ii) The data for this exercise are in the file `https://minerva.it.manchester.ac.uk/~saralees/geyser.txt` which comprises three columns headed `day`, `duration` and `interval`. We consider only the duration data in the second column, which correspond to the durations of $n = 222$ eruptions of the Old Faithful Geyser in Yellowstone National Park in the USA.

Explore these data in R using summary statistics, a histogram and box plot. What are the main features of the shape of the empirical distribution of the data? Do you think a single summary measure of location is adequate for these data? How might you summarise the location of these data?

(iii) The data are in the file `https://minerva.it.manchester.ac.uk/~saralees/anorexia.txt` comprising of four columns: `case`, `postwt`, `prewt` and `treatment`. These data consist of the weights, in kg, of girls receiving three different kinds of anorexia treatment over a fixed period of time. The controls (group 1) received the standard treatment, the second group (coded as 2) received cognitive behavioural treatment and the third group (coded as 3) received family therapy. The weight of each girl was measured before and after treatment and these are denoted `prewt` and `postwt`, respectively in the data file.

The objective is to compare the effects of the three treatments. To facilitate this, create a new variable recording the change in weight for each girl:

```
> dwt <- postwt-prewt
```

Additional vectors containing the differences for each treatment group on its own, can be constructed, for example, via:

```
> std.dwt <- dwt[treatment==1]
```

which in this case will create a vector called `std.dwt` containing the `dwt` values for those receiving treatment 1.

Use summary statistics, histograms and box plots to explore differences between the effects of the three treatments. Which treatment (if any) do you conclude works best and which (if any) is the least effective?