A9: Statistics – Getting started with R

Installing R

R is a free software environment for statistical computing and graphics.

To install R, go to `https://cran.r-project.org` and choose your operating system, download and install.

Installing RStudio

I recommend that you use RStudio to work with R. RStudio is a free integrated development environment (IDE) for R. You can use RStudio to enter code, read in datasets, store your work and make plots – all within a single environment.

R and RStudio are installed separately.

To install RStudio, go to `https://posit.co/download/rstudio-desktop/` and download/install the free desktop version of RStudio for your operating system.

Using R as a calculator

Run RStudio. The RStudio window you get is normally divided into four: a window where you can write and edit code (initially labelled "Untitled1", normally top left); the main console window where you can type code, or cut and paste it (bottom left); a window with environment/history tabs (top right); a window with files/plots/... tabs (bottom right), the plot tab is where your plots will appear.

Alternatively run R instead of RStudio.

You should have a "Console" window which contains a welcome message, and underneath the welcome message is a line like the following:

```
>
```

The `>` is the command prompt, indicating that R is ready for you to type a command. For example, typing `8+5` and then pressing the `Enter` key, you should see the following:

```
> 8+5
[1] 13
>
```

Don't type the `>` character, just type `8+5` then press `Enter`. R has computed 8+5. The last `>` indicates that R is ready for the next command. The first element of the answer is labelled `[1]` even when, as above, there is only one element.

In everything that follows, don't type the `>` character at the beginning of a line, R prints that to indicate that it is ready for new input from you.

To set up a vector `x` containing the elements 1.4, 3.5, 7.2, type the following, then press `Enter`.

```
x <- c(1.4, 3.5, 7.2)
```

The symbol `<-` (i.e. "less than", followed by "minus") is the assignment operator in R. The above sets `x` equal to the vector $(1.4, 3.5, 7.2)$. To check this, type `x` then press `Enter` (this prints `x`) and you should see

```
> x
[1] 1.4 3.5 7.2
```

You can also use `=` instead of `<-` for assignment, so `x = c(1.4, 3.5, 7.2)` has the same effect.

To generate a vector of integers from 1 to 5, a shortcut is to use `1:5` instead of `c(1, 2, 3, 4, 5)`.

```
y <- 1:5
```

The operation `y + 10` adds 10 to each component of `y`, and similarly `y / 8` divides each component of `y` by 8. Subtraction (e.g. `y - 20`) and multiplication (e.g. `4 * y`) work similarly. For example

```
> y + 10
[1] 11 12 13 14 15
```

The above does not change the value of `y`. To define `z` to be `y + 10` use

```
z <- y + 10
```

and to check the answer

```
> z
[1] 11 12 13 14 15
```

Q-Q plots

To randomly generate a sample of size 100 from four different densities, from (i) a $N(0,1)$ density, (ii) an exponential (parameter 1) density, (iii) a Uniform$(0,1)$ density, and (iv) a t-density with 1 degree of freedom, we can use the following.

```
x1 <- rnorm(100)
x2 <- rexp(100)
x3 <- runif(100)
x4 <- rt(100, df = 1)
```

The `r` at the beginnining of each of the function-names `rnorm`, `rexp`, `runif`, `rt` signifies that we are asking for a random sample from densities (i)–(iv). (Type `?rnorm` to see the detailed help page for the `rnorm` function, similarly use `?rexp` and so on.)

We can then do normal Q-Q plots (one Q-Q plot for each sample) using

```
qqnorm(x1)
qqnorm(x2)
qqnorm(x3)
qqnorm(x4)
```

After running each of these commands, you should see a new Q-Q plot. The ordered sample is on the vertical axis, e.g. the ordered values of the vector `x1` in the first case. The ordered sample values are plotted against the values of $\Phi^{-1}(\frac{k}{n+1})$ for $k = 1, \ldots, n$ for a normal Q-Q plot, where here $n = 100$.

Can you explain the shape of the four plots? (See the corresponding question on Sheet 1.)

The `precip` dataset is part of R. Type `?precip` to see a brief description of it. For a normal Q-Q plot of the dataset, we can use

```
qqnorm(precip)
```

Plotting a $N(0,1)$ density

We will plot a $N(0,1)$ density function. The form of the `plot` command is `plot(x, y)`, where `x` and `y` are vectors of numerical values (of equal lengths).

The first line below sets `x` equal to a vector of 101 elements, equally spaced between -4 and 4. The second line computes the corresponding vector of `y` values. The third line plots `y` against `x`, and the additional argument `type = "l"` says that the points should be joined by lines (rather than be plotted as separate points, which is what would happen if `type = "l"` was omitted).

```
x <- seq(from = -4, to = 4, length.out = 101)
y <- 1/sqrt(2 * pi) * exp(-x^2/2)
plot(x, y, type = "l")
```

Next steps

- Do the problem sheet questions involving R. R-code will be supplied to help you.
- The official introduction to R manual is at `https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf`. Although this probably contains more than you need, it might be useful for reference.
- The lecture notes for the statistical programming part of A12 contain plenty of examples and may also be useful, see `http://www.stats.ox.ac.uk/~evans/statprog/index.htm`.