

## Programming: practical 2a

Dr Colin Gillespie

IN THIS question, we are going to use a `for` statement to loop over a large data set and construct some scatter plots. To generate the data, run the following piece of R code

```
library("nclRprogramming")
data(dummy_data)
dd = dummy_data
```

The data frame `dd` represents an experiment, where we have ten treatments:  $A, B, \dots, J$  and measurements at some time points. We want to create a scatter plot of measurement against time, for each treatment type.

1. First we create a scatter plot of one treatment:

```
plot(dd[dd$treatments=="A",]$time,
     dd[dd$treatments=="A",]$measurement)
```

Since the colnames are a bit long, let's shorten them:

```
colnames(dd) = c("m", "t", "trts")
```

2. To generate a scatter-plot for each treatment, we need to iterate over the different treatment types:

```
for(treat in unique(dd$trts)) {
  plot(dd[dd$trts==treat,]$t,
       dd[dd$trts==treat,]$m)
  readline("Hit return for next plot\n")
}
```

A few questions:

- What does `unique(dd$trts)` give?
  - In the `for` loop, what variable is changing? What are its possible values?
  - What does the `readline` function do?
3. The default axis labels aren't great. So we can change the  $x$ -axis label using `xlab`:

```
plot(dd[dd$trts==treat,]$t,
     dd[dd$trts==treat,]$m,
     xlab="Time")
```

Use `ylab` to alter the  $y$ -axis label.

4. To add a title to a plot we use the `main` argument, viz:

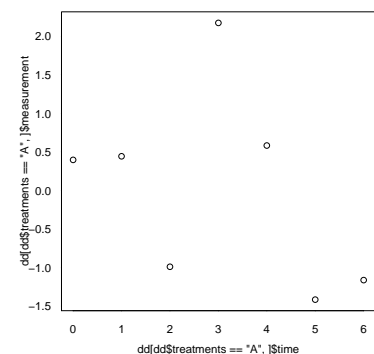


Figure 1: Measurements against time.

```
plot(dd[dd$trts=="A",]$t,
     dd[dd$trts=="A",]$m,
     main="Treatment",
     xlab="Time", ylab="Measurement")
```

We can combine strings/characters using the paste function,

```
paste("Treatment", treat)

## [1] "Treatment J"
```

Rather than have a static title, make the title of each plot display the treatment type.

5. The y-axis range should really be the same in all graphics. Add a ylim argument to fix the range.<sup>1</sup>
6. At each iteration, use the message function to print the average measurement level across all time points.
7. On each graph, highlight any observations with a blue point if they are larger than the mean + standard deviations or less than the mean - standard deviations. Use the points function to highlight a point.<sup>2</sup> For example, to highlight the points (1,2) and (3, 4) we use the command:

```
points(c(1, 3), c(2, 4), col = 2)
```

8. Suppose we wanted to save individual graphs. Add in the pdf function to save the resulting graph. To get unique file names, use the paste command:

```
filename = paste("file", treat, ".pdf", sep = "")
```

9. Put the your code, i.e. the for loop and plotting commands, in a function which takes the data frame as an argument.
10. Alter your function to take another argument where you can save the graph in a different directory.

*Final piece of code*

*Solutions*

Solutions are contained within this package:

```
library("nclRprogramming")
vignette("solutions2a", package = "nclRprogramming")
```

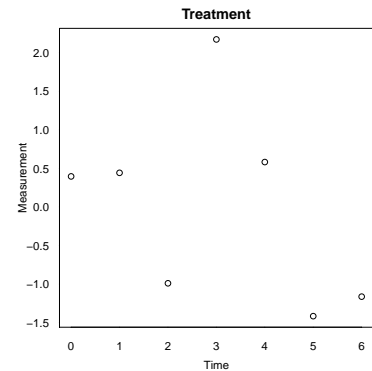


Figure 2: Measurements against time with a title.

<sup>1</sup> Hint: Work out the range before the for loop.

<sup>2</sup> Hint: You don't need if statements here. Just subset your data frame and pass this new data frame to the points function.