

Programming: Practical 2a solutions

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?¹
 - In the `for` loop, what variable is changing? What are its possible values?²
 - 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.

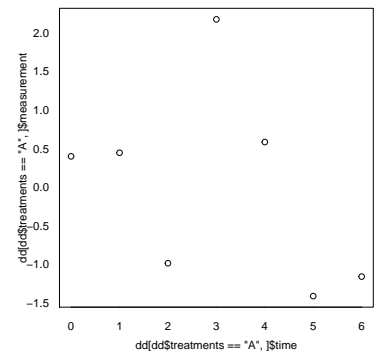


Figure 1: Measurements against time.

¹ It gives all treatments.

² The `treat` variable is changing. It goes through the different treatments.

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

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

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

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

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

5. The y-axis range should really be the same in all graphics. Add a ylim argument to fix the range.³

³ Hint: Work out the range before the for loop.

```
range(dd$m)

## [1] -1.639  8.113

plot(dd[dd$trts=="treat,]$t,
     dd[dd$trts=="treat,]$m,
     main=paste("Treatment", treat),
     xlab="Time", ylab="Measurement",
     ylim=c(-2, 10))
```

6. At each iteration, use the message function to print the average measurement level across all time points.

```
## Within the for loop have the line
message(mean(dd[dd$trts == treat, ]$m))

## 4.9469753296987
```

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.
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", "1", ".pdf", sep = ""))

## [1] "file1.pdf"
```

9. Put the above code 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

```
viewgraphs = function(dd, colour=TRUE, save=FALSE) {
  for(treat in unique(dd$trts)) {
    if(save) {
      filename = paste("file", treat, ".pdf", sep="")
      pdf(filename)
    }
    ##Use a different shape in the points
    if(colour) pch = 19
    else pch = 22
    ##Do selection one
    sel = (dd$trts == treat)
    plot(dd[sel,]$t, dd[sel,]$m,
          ylab=treat, xlab="Time",
          main=paste("Treatment", treat))
    ##Calculate the limits
    values = dd[sel,]$m
    message(mean(values))
    upper_lim = mean(values) + sd(values)
    lower_lim = mean(values) - sd(values)
    ##Extract the points
    up_row = dd[sel & dd$m > upper_lim,]
    low_row = dd[sel & dd$m < lower_lim,]
    ##pch=19 gives a solid dot
    ##See ?points
    points(up_row$t, up_row$m, col=4, pch=pch)
    points(low_row$t, low_row$m, col=4, pch=pch)
    if(save){
      dev.off()
    } else {
      readline("Hit return for next plot\n")
    }
  }
}
```