A short R heatmap tutorial

A.Bailey

29 July 2015

Contents

Introduction

This is a short tutorial for producing heatmaps in R using a modified data set provided by Leanne Wickens. I assume the reader is reasonably au fait with R Studio and able to install packages, load libraries etc..... in order to use this code. It should be possible to replicate this analysis exactly following the code below.

Note Use the question mark ? in front any function in R to see all the details of a function in the help file e.g. `?read.csv`. Or Google for examples on Stackoverflow.

The purpose of this analysis

Here we have a test data set of 63 proteins from 6 experiments where cells were either untreated (controls) or treated with a growth factor to examine the effect on protein expression. Hence the questions we seek to answer are:

- Can we identify and group proteins that either increase or decrease their expression when treated as compared with untreated cells?
- Is this consistent between experiments? i.e. Are the experiments reproducible?
- Can we visualise the analysis as a heatmap?

Prepare,load and preprocessing the data

The knitr settings for the R markdown file are to show the R code, cache the data and load the required packages and set the figure dimensions:

```
knitr::opts_chunk$set(cache=TRUE,echo=TRUE,cache=TRUE, message=FALSE,
                      warning=FALSE, fig=TRUE,eval=TRUE,
                      fig.width=9, fig.height = 9,fig.lp='fig:')
```

These are the two packages we need to load in addition to the base packages for performing the analysis

```
library(d3heatmap)
library(gplots)
```

For test data to load, here is the URL to Leanne's modified test data which contains anonymised observations for 63 proteins for three control experiments and three experiments where cells are treated with a growth factor.

```
fileUrl <- "https://github.com/ab604/heatmap/blob/master/leanne_testdata.csv"
```

If the data file `leanne_testdata.csv` is not already downloaded to your working directory, this is the code to get it. It also produces a log file:

```
if(!file.exists("leanne_testdata.csv")){
        download.file(fileUrl,destfile="leanne_testdata.csv")
        dateDownloaded <- date()
        log_con <- file("testdata_download.log")
        cat (fileUrl,"\n","destfile= leanne_testdata.csv",
             "\n","destdir =", getwd(),"\n",dateDownloaded,
             file = log_con)
        close(log_con)
}
```

Next we load the data into R. Here first I created some meaningful column names as `cols` for the three controls and three treated experiments respectively. These names are used when reading in the `.csv` file with `read.csv` to the `dat` data frame, by declaring `col.names = cols`. `row.names = 1` makes it read column one (and I don't need to rename this column) the protein accession codes, as row names - here these are just numbers from 1 to 63.

```
# Create columnn names for data
cols <- c("","Control.1","Control.2","Control.3","Treated.1",
          "Treated.2","Treated.3","Ttest")

# Load data, re-naming columns and taking column 1 for row names.
dat <- read.csv("leanne_testdata.csv", row.names = 1,
                stringsAsFactors = FALSE, col.names = cols, header = TRUE)
```

We can see how that looks by using the `head` command:

```
# Look at first five lines of dat
head(dat,5)
```

```
##   Control.1 Control.2 Control.3 Treated.1 Treated.2 Treated.3       Ttest
## 1  7.152522  7.163227  7.112041  6.912676  6.882109  6.918670 0.000412886
## 2  6.639397  6.614227  6.594811  6.365425  6.349997  6.411068 0.000741591
## 3  3.678131  3.776133  3.813182  2.398460  2.088798  2.172404 0.001009950
## 4  5.040845  5.007151  4.686440  3.425154  3.516301  3.657402 0.001262187
## 5  6.915152  6.810610  6.897310  7.124722  7.211372  7.269469 0.004306239
```

We don't want the Ttest column, so let's sort that out next.

Tidying and normalising the data

I then extract columns 1 to 6 to a new data frame called `dat.tdy` as we don't need the last column.

```
# Tidy the data, extract only columns 1 to 6
dat.tdy <- dat[,1:6]

# Look at first 5 lines of the tidy data
head(dat.tdy,5)
```

```
##   Control.1 Control.2 Control.3 Treated.1 Treated.2 Treated.3
## 1  7.152522  7.163227  7.112041  6.912676  6.882109  6.918670
## 2  6.639397  6.614227  6.594811  6.365425  6.349997  6.411068
## 3  3.678131  3.776133  3.813182  2.398460  2.088798  2.172404
## 4  5.040845  5.007151  4.686440  3.425154  3.516301  3.657402
## 5  6.915152  6.810610  6.897310  7.124722  7.211372  7.269469
```

We normalise the data using `scale`, transposing `dat.tdy` using `t` to perform row-wise normalisation so that: $z = \frac{x-\mu}{\sigma}$ to create `dat.n`

Additionally, I've transposed the data back to the original form as `dat.tn`, for use later on.

```
# Normalize tidy data, transpose for row wise normalisation
dat.n <- scale(t(dat.tdy))

# Put data back in original form
dat.tn <- t(dat.n)
```

Just for completeness, to check the normalisation, here we calculate the means and standard deviation for each row (each protein):

```
# Check means are zero and std devs are 1
round(colMeans(dat.n),1)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
##  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
##  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 51 52 53 54 55 56 57 58 59 60 61 62 63
##  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
apply(dat.n,2,sd)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
##  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
##  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## 51 52 53 54 55 56 57 58 59 60 61 62 63
##  1  1  1  1  1  1  1  1  1  1  1  1  1
```

Hierarchical clustering

Next we do the hierarchical clustering needed for the heatmap. We could just let the heatmap function do the work, but here we'll do it step by step to understand what is going on.

Relating data points as distances

So we have our data here as concentrations, but how do we group them? The simplest thing to do is to turn the data into distances, as a measure of similarity, where close things are similar and distant things are dissimilar.

The Euclidean distance d between a pair of observations x_i and y_i is defined as:

$d = \sqrt{\sum_i (x_i - y_i)^2}$

So lets calculate the distance between the rows in `dat.n`. In `dat.n` the experiments are in the rows. This calculates is distance is between the experiments for all the proteins in each experiment. What would we expect?

We'd expect the controls to be close to each other and the treated to be close to each other, right?

Let's do this in detail, for example the distance between `Control.1` and `Control.2` is `sqrt(sum((dat.n[1,]-dat.n[2,])^2))`.

This means we take the row 2 values from row 1 values, squaring the results and summing them all to a single value and taking the square root to find the linear distance between these rows, which is 6.427.

You can check this against the first value in `d1` that we calculate below in using `dist`.

```
# Calculate distance between experiments in rows
d1 <- dist(dat.n,method = "euclidean", diag = FALSE, upper = FALSE)

# Show distances
round(d1,3)
```

```
##           Control.1 Control.2 Control.3 Treated.1 Treated.2
## Control.2     6.427
## Control.3     5.969     5.226
## Treated.1    15.446    12.133    13.369
## Treated.2    16.094    12.951    14.641     5.151
## Treated.3    14.411    11.300    12.646     5.095     5.685
```

As we expected, controls are closer to each other than treated and vice versa. Then we cluster according to these distances.

Let's do the same for the rows in `dat.tn` where the rows are the proteins and the columns are experiments. This time we're calculating how similar each protein is to each other across all six experiments. We don't know what to expect here, but the principle is the same.

```
# Calculate distance between proteins in rows
d2 <- dist(dat.tn,method = "euclidean", diag = FALSE, upper = TRUE)
```

So now we have two distance matrices, one measuring experimental similarity and the measuring protein similarity between the experiments. But it would be more useful to cluster these results and then compare them, so let's do that now.

Notes

- We could have done this just using `dat.n` and transposed it when we used `dist` to obtain the `d2` distance matrix.
- We had to make two distance matrices here as we didn't have a square matrix i.e. the number of rows and columns aren't equal in `dat`. If they were square we could just make one and only do one clustering.

Clustering

There are lots of flavours of clustering, and no clear way to say which is best. Here we'll use the Ward criterion for clustering which attempts to minimise the variance within clusters as it merges the data into clusters, using the distances we've calculated. The data is merged from the bottom up (aka agglomeration) adding data points to a cluster and splitting them according to the variance criterion. See Wikipedia for more detail: Hierarchical clustering

We'll cluster both distance matrices `d1` and `d2` here using `hclust`.

```
# Clustering distance between experiments using Ward linkage
c1 <- hclust(d1, method = "ward.D2", members = NULL)
# Clustering distance between proteins using Ward linkage
c2 <- hclust(d2, method = "ward.D2", members = NULL)
```

Now lets look at the dendrograms made by clustering our distance matrices `d1` and `d2` using the code below:

```
# Check clustering by plotting dendrograms
par(mfrow=c(2,1),cex=0.5) # Make 2 rows, 1 col plot frame and shrink labels
plot(c1); plot(c2) # Plot both cluster dendrograms
```

Again, as we'd expect, the controls and treatments to cluster respectively.

Creating a heatmap from both clustering solutions

We can now use our clustering solutions to make a heatmap. Here I used `heatmap.2` and provide the code to make an optional interactive HTML heatmap using `d3heatmap`.

The following code plots the tidy, normalised data in `dat.tn`, arranged column wise according to the experiments clusters `c1` and the protein clusters `c2` row wise.

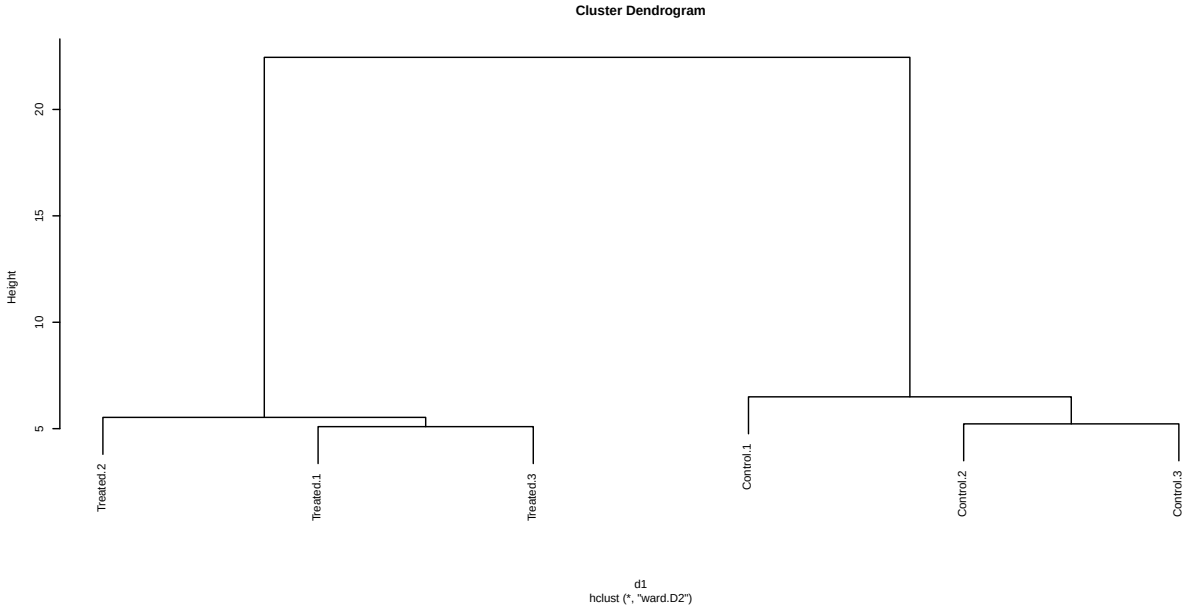
This means that we can compare experimental differences for each protein and between control and treatment in a single plot.

I also set my colour palette as blue, white and red for the heatmap for as `my_palette` with 25 increments, but you can change this to whatever you wish.

Here red indicates an increase in a proteins expression compared the mean expression for that protein, and blue a decrease in expression compared to the mean. So we can clearly see which proteins increase and decrease their expression in cells that have been treated as compared with controls, and see for which proteins this is consistent across all experiments.

```
# Set colours for heatmap, 25 increments
my_palette <- colorRampPalette(c("blue","white","red"))(n = 25)

# Plot heatmap with heatmap.2
par(cex.main=0.75) # Shrink title fonts on plot
heatmap.2(dat.tn,                       # Tidy, normalised data
          Colv=as.dendrogram(c1),       # Experiments clusters in cols
          Rowv=as.dendrogram(c2),       # Protein clusters in rows
          density.info="histogram",     # Plot histogram of data and colour key
          trace="none",                 # Turn of trace lines from heat map
          col = my_palette,             # Use my colour scheme
          cexRow=0.5,cexCol=0.75)       # Amend row and column label fonts
```


Cluster Dendrogram
Height
5
10
15
20
Treated.2
Treated.1
Treated.3
Control.1
Control.2
Control.3
d1
hclust (*, "ward.D2")

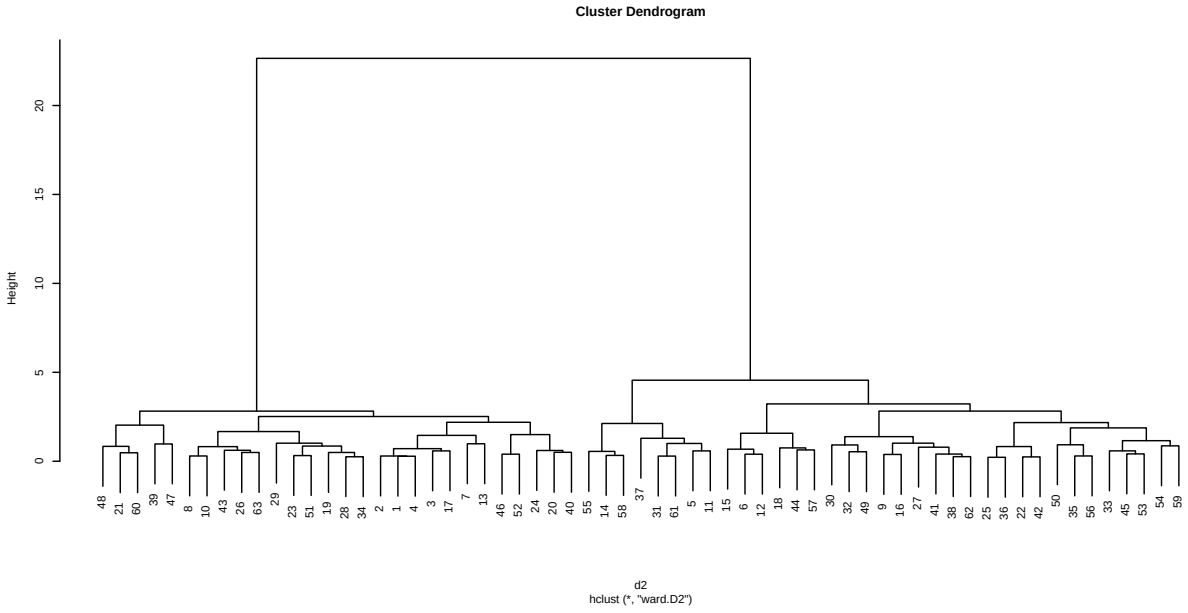

Cluster Dendrogram
Height
0
5
10
15
20
48 21 60 39 47 8 10 43 26 63 29 23 51 19 28 34 2 1 4 3 17 7 13 46 52 24 20 40 55 14 58 37 31 61 5 11 15 6 12 18 44 57 30 32 49 9 16 27 41 38 62 25 36 22 42 50 35 56 33 45 53 54 59
d2
hclust (*, "ward.D2")

Figure 1

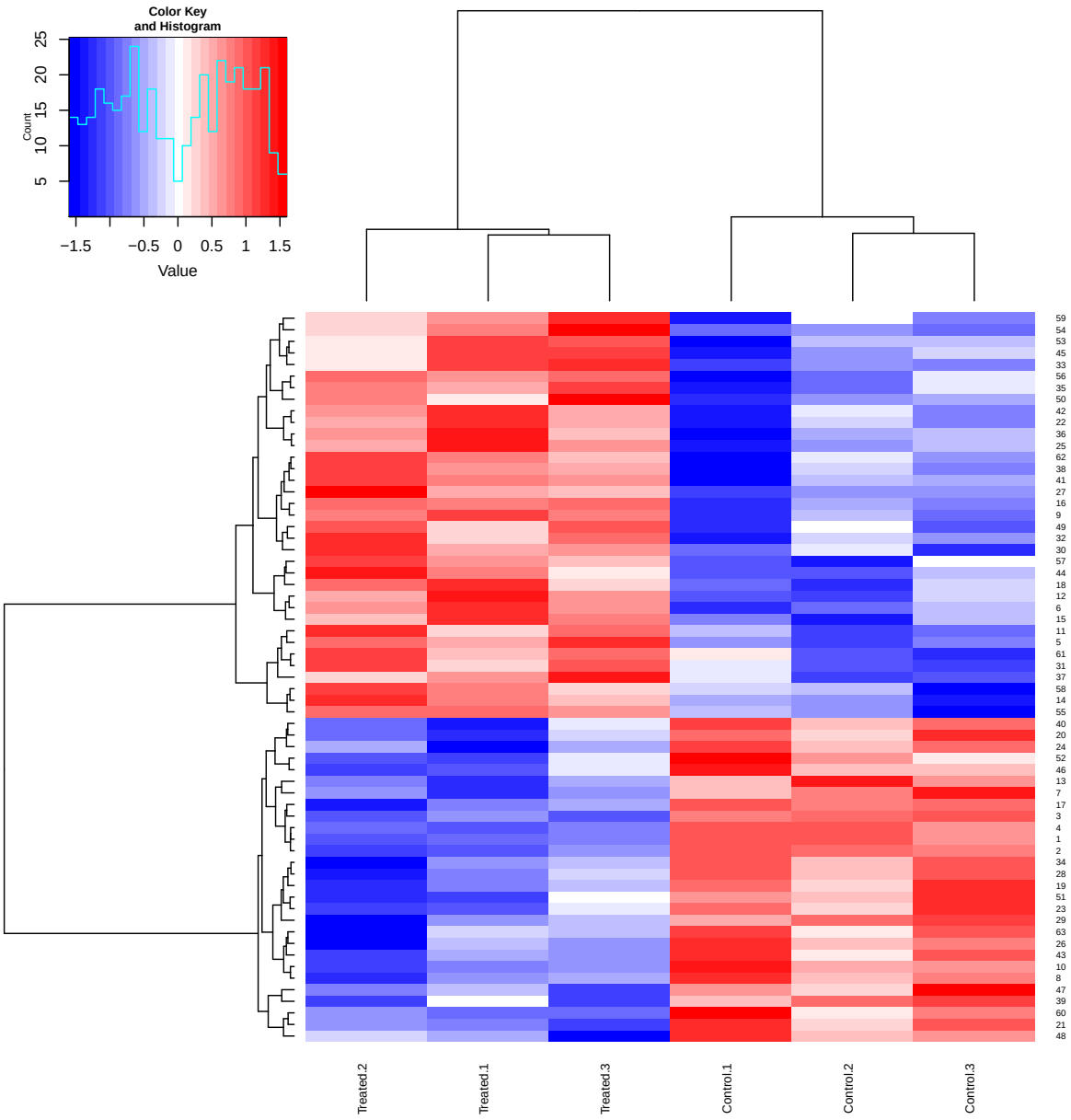


Figure 2: Heatmap of test data

Extras

Below is code for outputting the heatmap to a `.pdf`, and to `.png` for Word or Power-point.

```
# Plot heatmap with heatmap.2

# Open a file for pdf
pdf(file="my_heatmap.pdf",paper = "a4")
par(cex.main=0.75) # Shrink title fonts
heatmap.2(dat.tn, Colv=as.dendrogram(c1), Rowv=as.dendrogram(c2),
          density.info="histogram", trace="none",col = my_palette,
          cexRow=0.5,cexCol=0.75)
# Close file
dev.off()

# Open a file for a png for Word
png(file="my_heatmap.png", width = 800, height = 800, units = "px", bg="white")
par(cex.main=0.75) # Shrink title fonts
heatmap.2(dat.tn, Colv=as.dendrogram(c1), Rowv=as.dendrogram(c2),
          density.info="histogram", trace="none",col = my_palette,
          cexRow=0.5,cexCol=0.75)
# Close file
dev.off()
```

Below is code to make an interactive heatmap for a webpage.

```
# Create an interactive HTML heatmap with d3heatmap
d3heatmap(dat.tn,Rowv=as.dendrogram(c2),Colv=as.dendrogram(c1),col=my_palette)
```