THE UNIVERSITY *of* EDINBURGH
School of Biological Sciences

Assignment

Using R for Data Science

Student Exam Number: B196466

The file name of your uploaded document **must** include your exam number

Assignment

B196466

2021/11/28

Guideline

Often when working with data we are interested in using R to build data visualisations. We have data for the processed numerical data for plotting. We have annotation data that will allow us to build gene and sample names. Each row in the annotation has a gene name and each row has a row name, as well as other information. We have a list which is a collection of row names for each element we need to plot. Each row in the processed numerical data has a unique identifier-the row name. This can be used to uniquely index each row in R. The elements of the gene list match the row name in the data table and the same gene listed more than once in the gene list. Not all Genes have a Gene name but all have a unique identifier.

First part: Accessing the Data

- Firstly, we need to create a directory and change our working directory there.
- Then copy the file which we need to our working directory and unzip it.
- In terminal, For example, like the following:
- mkdir assignment
- cd assignment
- cp -R /shared_files/RDS_assignment/s2170612_files.zip .
- unzip s2170612_files.zip
- So now we have the information about the files that we need to make the plot. We can then bring the data into R.
- And in R, we need to input "setwd("~/assignment")" to set it as our working directory.

```
#Note: We can use ls() and dir() to check the files in the workspace.
dir()
```

```
## [1] "Assignment.R"          "Assignment.Rmd"        "data_all.csv"
## [4] "gene_annotation.csv"   "genelist_s2170612.txt" "sample_annotation.csv"
```

Second part: Loading Data into R.

- Note: If you put the files in another location, please adjust the paths
- Note: We are now using relative paths so if the data folder is moved all the code will still work!
- Note: We can use head() and tail() command for checking tables.
- Note: We can use class() and typeof() to check the types of the objects created.

The input data table which called "data_all.csv" is numerical data with row 1 containing sample names and column 1 containing gene names.

```
#read file "data_all.csv"
data<-read.csv("data_all.csv",header = T,row.names = 1)
```

```
#check the file result
head(data)
```

```
##          A        B        C        D        E        F        G        H
## 1 6.242349 6.302403 6.584037 6.645033 6.591126 6.536004 4.465151 4.175000
## 2 5.404286 5.667919 5.731933 8.686235 8.841934 8.728940 8.998052 8.436326
## 3 2.648436 2.685151 2.727255 2.518935 2.412016 2.422645 2.345909 2.284605
## 4 9.694978 9.164112 9.415601 5.111681 5.769811 5.322498 4.300614 4.387367
## 5 6.760503 6.474128 6.503369 6.847321 7.281944 7.139831 5.739700 6.244461
## 6 2.590525 2.397822 2.799170 2.575602 2.760729 2.795382 2.416251 2.480369
##          I        J        K        L
## 1 4.587410 2.813028 2.597596 2.805376
## 2 8.691272 3.834652 3.632691 3.741054
## 3 2.514958 5.299666 5.340036 5.652512
## 4 4.244163 4.122095 3.983176 4.092571
## 5 6.519948 3.134813 3.793360 3.437815
## 6 2.797060 4.935703 4.655196 5.535270
```

```
#check the types of the objects created
class(data)
```

```
## [1] "data.frame"
```

There is a table "gene_annotation.csv" containing gene annotation.

```
#read file "gene_annotation.csv"
gene<-read.csv("gene_annotation.csv")
#check the file result
head(gene)
```

```
##   X Gene Type   LongName
## 1 1    1   XA 1436799_at
## 2 2    2   XA 1436227_at
## 3 3    3   XA 1420504_at
## 4 4    4   XA 1417945_at
## 5 5    5   XA 1420337_at
## 6 6    6   XA 1439944_at
```

```
#check the types of the objects created
class(gene)
```

```
## [1] "data.frame"
```

There is a table "sample_annotation.csv" containing sample annotation.

```
#read file "sample_annotation.csv"
sample<-read.csv("sample_annotation.csv")
#check the file result
head(sample)
```

```
##   X SampleName TreatmentGroup
## 1 1          A              1
## 2 2          B              1
## 3 3          C              1
## 4 4          D              2
## 5 5          E              2
## 6 6          F              2
```

```
#check the types of the objects created
class(sample)
```

```
## [1] "data.frame"
```

The last file provides a list of genes to use for plotting which called "genelist_s2170612.txt".

```
#read file "genelist_s2170612.txt"
genelist<-read.csv("genelist_s2170612.txt")
#check the file result
head(genelist)
```

```
##     x
## 1   1
## 2 176
## 3 135
## 4  60
## 5 119
## 6  79
```

```
#check the types of the objects created
class(genelist)
```

```
## [1] "data.frame"
```

Third part: De-duplicate genelist name

Select the required data for the plot and only those genes on the "genelist_s2170612.txt" need to be plotted. Each gene should only appear once on the final plot for some of the genelist ID are not unique.we can remove the `duplicate()` elements using the command duplicated()

```
#show duplicated genelist and set as "index"
index<-duplicated(genelist$"x")
```

```
#check the contents of genelist$"x"
print(genelist$"x")
```

```
##  [1]   1 176 135  60 119  79  31 168  17 115 198  91 193 100 143  18  55  26 155
## [20]  93 116 107 184  68 195  73  15  38  62 153  10  21  25  48  69 183  90  98
## [39] 120  52  93
```

```
#check the types of the objects created
class(genelist$"x")
```

```
## [1] "integer"
```

It returns a logical vector TRUE, FALSE, TRUE that indicates which elements are unique. Then just remove the FALSE elements by indexing with the logical vector

Use to "index" to remove duplicated genelist name in the file "genelist_s2170612.txt" and set as "new_genelist". Notice here we use the "!" operator to reverse the logical vector

```
new_genelist<-genelist[!index,]
```

```
#check the contents of index
print(index)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [37] FALSE FALSE FALSE FALSE  TRUE
```

Now we can see that there is one gene duplicated and it is the last one.Take care otherwise we keep the duplicates and discard the unique IDs!

```
#check the contents of new_genelist
head(new_genelist)
```

```
## [1]   1 176 135  60 119  79
```

Use new_genelist as an index to pull out only those elements needed for printing and set as "new_data"

```
new_data<-data[new_genelist,]
```

```
#check the type of new_data
class(new_data)
```

```
## [1] "data.frame"
```

Fourth part: plot annotate

Note: We can also use the rownames() and colnames() to check its rownames and colnames.

annotate each gene row with the type of gene (XA, XB or XC)

```
#set the rownames
rownames(gene)<- gene$LongName
```

```
#check the rownames of gene
head(rownames(gene))
```

```
## [1] "1436799_at" "1436227_at" "1420504_at" "1417945_at" "1420337_at"
## [6] "1439944_at"
```

change "Type" as string style

```
row_annotation=as.data.frame(gene['Type'])
```

```
#check the type of row_annotation
class(row_annotation)
```

```
## [1] "data.frame"
```

annotate each sample with the treatment group

```
#set the rownames
rownames(sample)<- sample$SampleName
```

```
#check the rownames of sample
head(rownames(sample))
```

```
## [1] "A" "B" "C" "D" "E" "F"
```

change "TreatmentGroup" as string style

```
col_annotation=as.data.frame(sample['TreatmentGroup'])
```

```
#check the type of col_annotation
class(col_annotation)
```

```
## [1] "data.frame"
```

Fifth part: Rename the gene names

Rename the gene names with the "LongName" from the gene annotation table. Use new_genelist as an index to pull out only those elements needed for printing and set as "new_gene".

```
new_gene<-gene[new_genelist,]
```

set the rownames

```
rownames(new_data)=new_gene$LongName
```

```
#check the rownames of data
head(rownames(new_data))
```

```
## [1] "1436799_at"   "1450082_s_at" "1435397_at"   "1430596_s_at" "1434291_a_at"
## [6] "1449154_at"
```

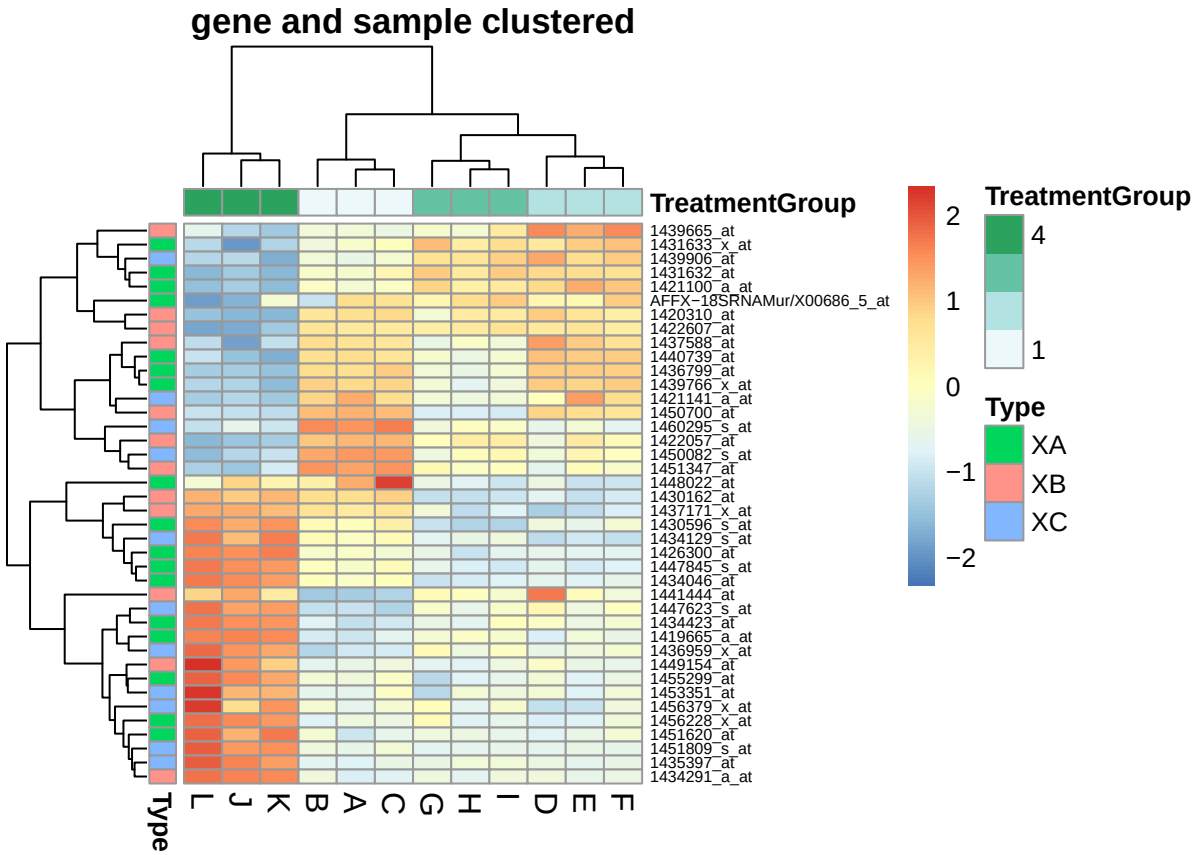
Sixth part: Create two heatmaps

quote "pheatmap" function

```
library(pheatmap)
```

pheatmap which both the genes and the samples are clustered

```
pheatmap(new_data,annotation_col=col_annotation,annotation_row=row_annotation,
         fontsize_row= 6,fontsize_col=12,scale='row',main='gene and sample clustered')
```

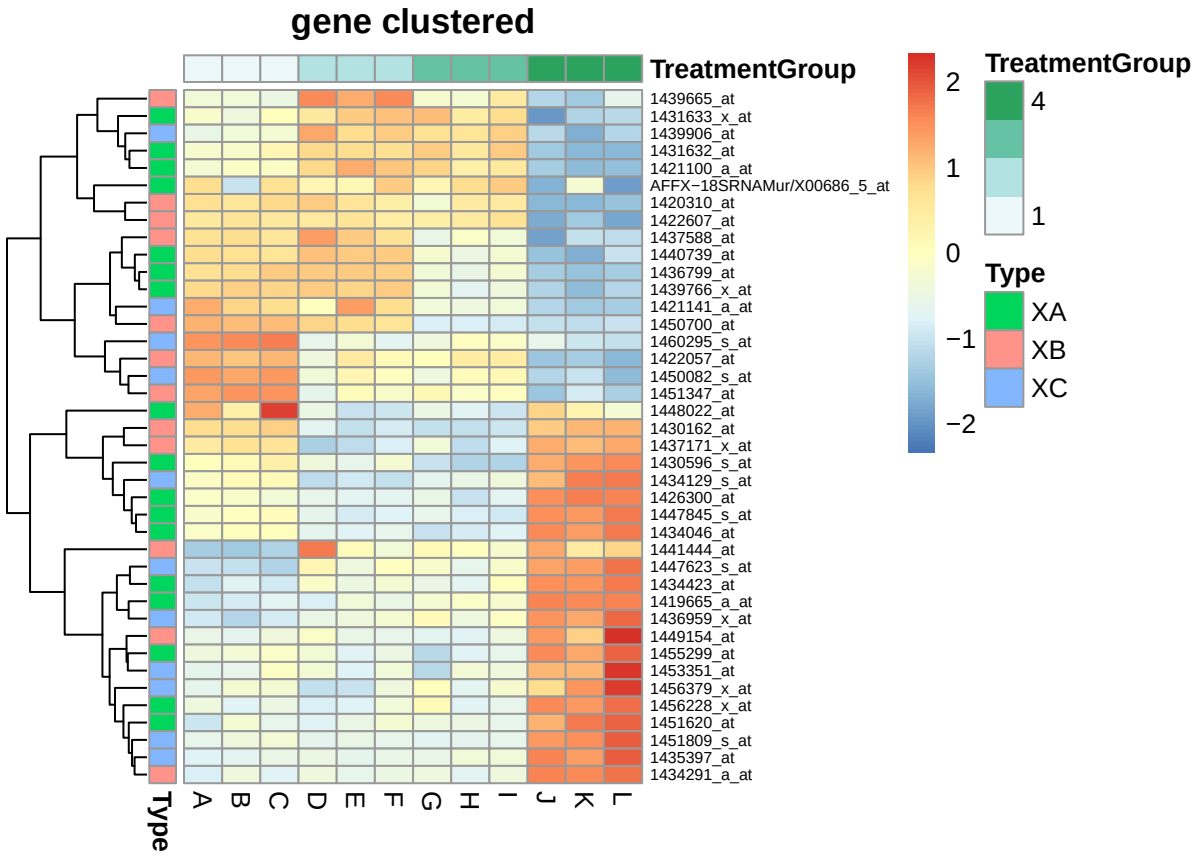


`annotation_col` means column annotation, `annotation_row` means row annotation,`fontsize_row` means

row size,`fontsize_col`means column size, `scale='row'`means parameters to normalise rows and `main`means title.

pheatmap which only the genes are clustered

```
pheatmap(new_data,annotation_col=col_annotation,annotation_row=row_annotation,
         fontsize_row =6,forntsize_col=12,scale='row',main='gene clustered',cluster_cols=FALSE)
```



Visual and aesthetic presentation of expression levels of multiple genes in multiple samples by using colour. Use colour gradients and similarity to reflect similarities and differences in the data. Cluster analysis essentially uses the degree of difference or similarity between two groups of values as a basis for clustering multiple groups of values in a hierarchical manner in order to ultimately obtain the clustering distances between samples.

We can see that “Type” and “TreatmentGroup” are very useful annotations. 1.Different “Treatment” change the expression level of different gene samples which increases some genes, decreases others. The Treatment 4 has the maximum impact and the treatment 1,2,3 have the similar result 2.Some genes with similar function cluster together for example(1440739_at and 1436799_at) 3.In the treatment 1,2,3, the expression level of Type XB gene is higher than that of Type XA and XC gene. 4. In the treatment 4, the expression level of Type XA gene is higher than that of Type XB and XC gene.