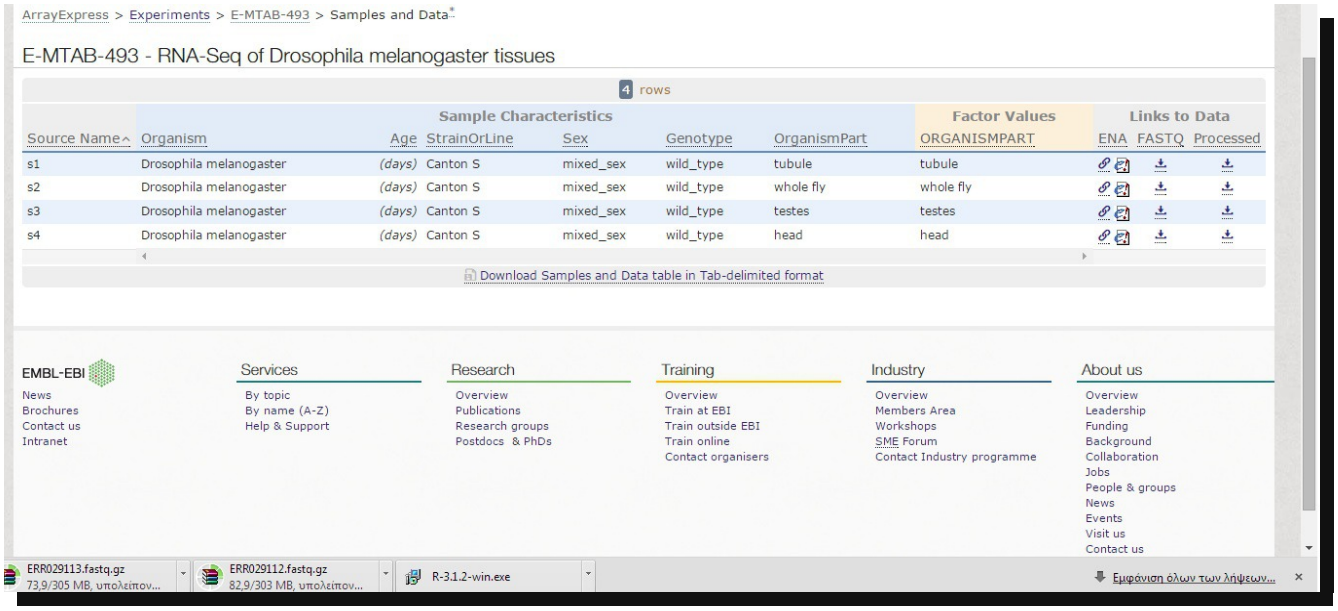
Final project for the course HY350.90

Topic 1: Gene Expression Analysis from microarray data

Task 1

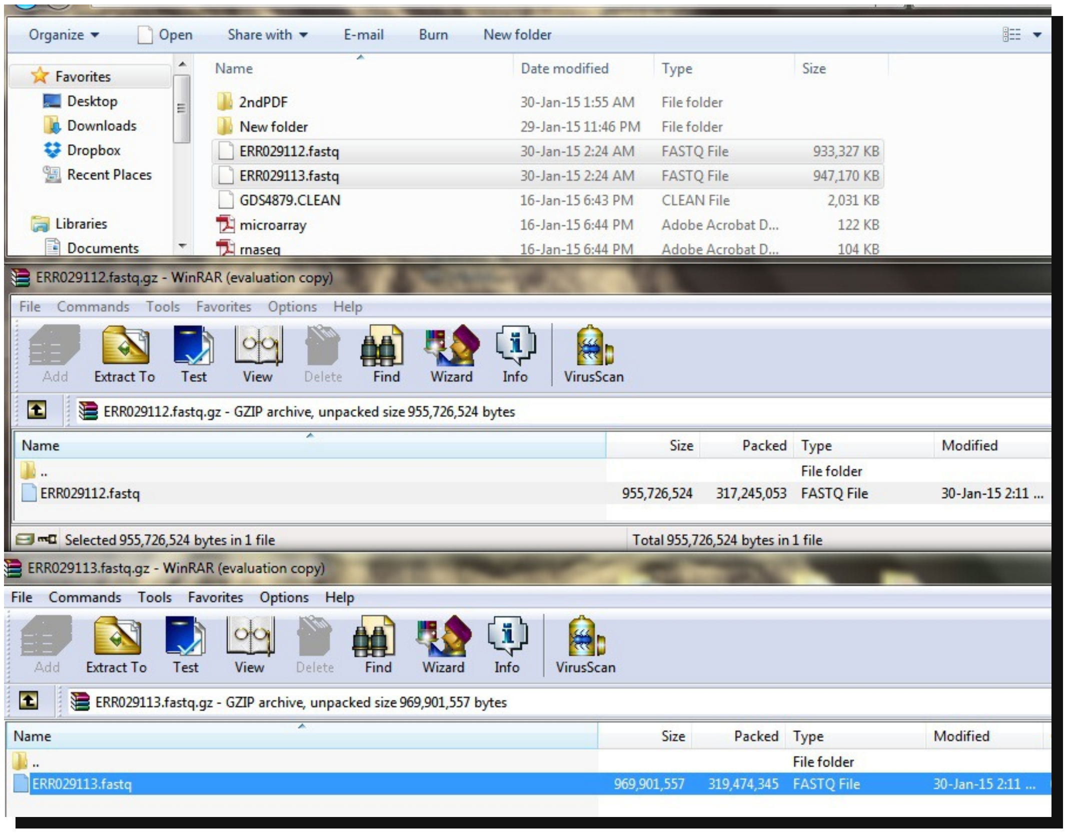
1. Find and download the dataset entitled
"Chronic high-level alcohol consumption effect on brain: post-mortem hippocampus".
Link: http://www.ncbi.nlm.nih.gov/sites/GDSbrowser?acc=GDS4879



E-MTAB-493 - RNA-Seq of Drosophila melanogaster tissues

	Sample Characteristics						Factor Values	Links to Data		
Source Name	Organism	Age	StrainOrLine	Sex	Genotype	OrganismPart	ORGANISMPART	ENA	FASTQ	Processed
s1	Drosophila melanogaster	(days)	Canton S	mixed_sex	wild_type	tubule	tubule			
s2	Drosophila melanogaster	(days)	Canton S	mixed_sex	wild_type	whole fly	whole fly			
s3	Drosophila melanogaster	(days)	Canton S	mixed_sex	wild_type	testes	testes			
s4	Drosophila melanogaster	(days)	Canton S	mixed_sex	wild_type	head	head			

Download Samples and Data table in Tab-delimited format



Name	Date modified	Type	Size
2ndPDF	30-Jan-15 1:55 AM	File folder	
New folder	29-Jan-15 11:46 PM	File folder	
ERR029112.fastq	30-Jan-15 2:24 AM	FASTQ File	933,327 KB
ERR029113.fastq	30-Jan-15 2:24 AM	FASTQ File	947,170 KB
GDS4879.CLEAN	16-Jan-15 6:43 PM	CLEAN File	2,031 KB
microarray	16-Jan-15 6:44 PM	Adobe Acrobat D...	122 KB
rnaseq	16-Jan-15 6:44 PM	Adobe Acrobat D...	104 KB

ERR029112.fastq.gz - WinRAR (evaluation copy)

ERR029112.fastq.gz - GZIP archive, unpacked size 955,726,524 bytes

Name	Size	Packed	Type	Modified
..			File folder	
ERR029112.fastq	955,726,524	317,245,053	FASTQ File	30-Jan-15 2:11 ...

Selected 955,726,524 bytes in 1 file   Total 955,726,524 bytes in 1 file

ERR029113.fastq.gz - WinRAR (evaluation copy)

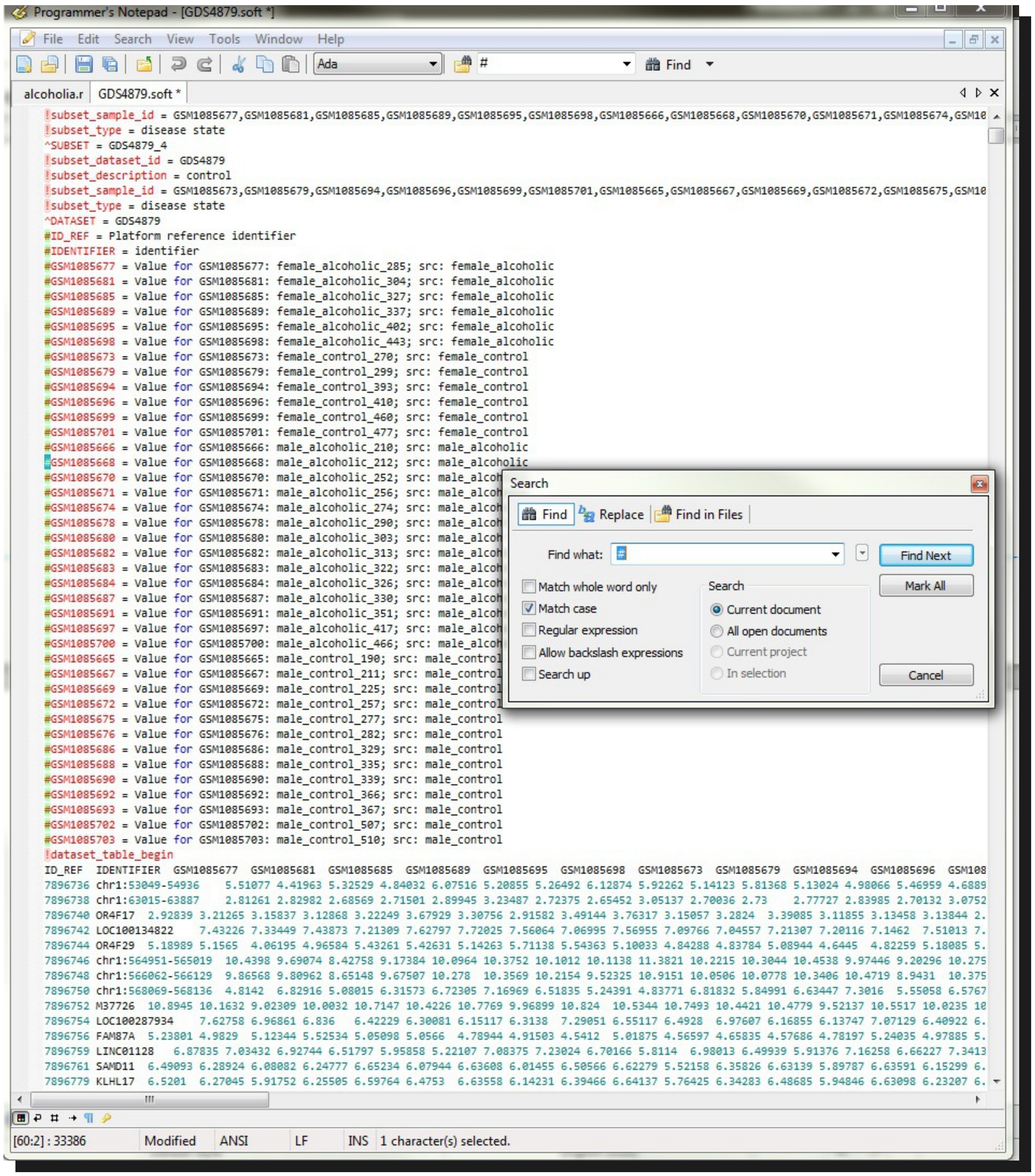
ERR029113.fastq.gz - GZIP archive, unpacked size 969,901,557 bytes

Name	Size	Packed	Type	Modified
..			File folder	
ERR029113.fastq	969,901,557	319,474,345	FASTQ File	30-Jan-15 2:11 ...

2. Clean the dataset from the unecessary lines in the beginning and the end.

These lines start with either of the symbols: # ! ^

Open file with a notepad and search for the characters :



```
!subset_sample_id = GSM1085677,GSM1085681,GSM1085685,GSM1085689,GSM1085695,GSM1085698,GSM1085666,GSM1085668,GSM1085670,GSM1085671,GSM1085674,GSM10
!subset_type = disease state
^SUBSET = GDS4879_4
!subset_dataset_id = GDS4879
!subset_description = control
!subset_sample_id = GSM1085673,GSM1085679,GSM1085694,GSM1085696,GSM1085699,GSM1085701,GSM1085665,GSM1085667,GSM1085669,GSM1085672,GSM1085675,GSM10
!subset_type = disease state
^DATASET = GDS4879
#ID_REF = Platform reference identifier
#IDENTIFIER = identifier
#GSM1085677 = Value for GSM1085677: female_alcoholic_285; src: female_alcoholic
#GSM1085681 = Value for GSM1085681: female_alcoholic_304; src: female_alcoholic
#GSM1085685 = Value for GSM1085685: female_alcoholic_327; src: female_alcoholic
#GSM1085689 = Value for GSM1085689: female_alcoholic_337; src: female_alcoholic
#GSM1085695 = Value for GSM1085695: female_alcoholic_402; src: female_alcoholic
#GSM1085698 = Value for GSM1085698: female_alcoholic_443; src: female_alcoholic
#GSM1085673 = Value for GSM1085673: female_control_270; src: female_control
#GSM1085679 = Value for GSM1085679: female_control_299; src: female_control
#GSM1085694 = Value for GSM1085694: female_control_393; src: female_control
#GSM1085696 = Value for GSM1085696: female_control_410; src: female_control
#GSM1085699 = Value for GSM1085699: female_control_460; src: female_control
#GSM1085701 = Value for GSM1085701: female_control_477; src: female_control
#GSM1085666 = Value for GSM1085666: male_alcoholic_210; src: male_alcoholic
#GSM1085668 = Value for GSM1085668: male_alcoholic_212; src: male_alcoholic
#GSM1085670 = Value for GSM1085670: male_alcoholic_252; src: male_alcoh
#GSM1085671 = Value for GSM1085671: male_alcoholic_256; src: male_alcoh
#GSM1085674 = Value for GSM1085674: male_alcoholic_274; src: male_alcoh
#GSM1085678 = Value for GSM1085678: male_alcoholic_290; src: male_alcoh
#GSM1085680 = Value for GSM1085680: male_alcoholic_303; src: male_alcoh
#GSM1085682 = Value for GSM1085682: male_alcoholic_313; src: male_alcoh
#GSM1085683 = Value for GSM1085683: male_alcoholic_322; src: male_alcoh
#GSM1085684 = Value for GSM1085684: male_alcoholic_326; src: male_alcoh
#GSM1085687 = Value for GSM1085687: male_alcoholic_330; src: male_alcoh
#GSM1085691 = Value for GSM1085691: male_alcoholic_351; src: male_alcoh
#GSM1085697 = Value for GSM1085697: male_alcoholic_417; src: male_alcoh
#GSM1085700 = Value for GSM1085700: male_alcoholic_466; src: male_alcoh
#GSM1085665 = Value for GSM1085665: male_control_190; src: male_control
#GSM1085667 = Value for GSM1085667: male_control_211; src: male_control
#GSM1085669 = Value for GSM1085669: male_control_225; src: male_control
#GSM1085672 = Value for GSM1085672: male_control_257; src: male_control
#GSM1085675 = Value for GSM1085675: male_control_277; src: male_control
#GSM1085676 = Value for GSM1085676: male_control_282; src: male_control
#GSM1085686 = Value for GSM1085686: male_control_329; src: male_control
#GSM1085688 = Value for GSM1085688: male_control_335; src: male_control
#GSM1085690 = Value for GSM1085690: male_control_339; src: male_control
#GSM1085692 = Value for GSM1085692: male_control_366; src: male_control
#GSM1085693 = Value for GSM1085693: male_control_367; src: male_control
#GSM1085702 = Value for GSM1085702: male_control_507; src: male_control
#GSM1085703 = Value for GSM1085703: male_control_510; src: male_control
!dataset_table_begin
ID_REF  IDENTIFIER  GSM1085677  GSM1085681  GSM1085685  GSM1085689  GSM1085695  GSM1085698  GSM1085673  GSM1085679  GSM1085694  GSM1085696  GSM108
7896736 chr1:53049-54936    5.51077 4.41963 5.32529 4.84032 6.07516 5.20855 5.26492 6.12874 5.92262 5.14123 5.81368 5.13024 4.98066 5.46959 4.6889
7896738 chr1:63015-63887    2.81261 2.82982 2.68569 2.71501 2.89945 3.23487 2.72375 2.65452 3.05137 2.70036 2.73    2.77727 2.83985 2.70132 3.0752
7896740 OR4F17  2.92839 3.21265 3.15837 3.12868 3.22249 3.67929 3.30756 2.91582 3.49144 3.76317 3.15057 3.2824  3.39085 3.11855 3.13458 3.13844 2.
7896742 LOC100134822    7.43226 7.33449 7.43873 7.21309 7.62797 7.72025 7.56064 7.06995 7.56955 7.09766 7.04557 7.21307 7.20116 7.1462  7.51013 7.
7896744 OR4F29  5.18989 5.1565  4.06195 4.96584 5.43261 5.42631 5.14263 5.71138 5.54363 5.10033 4.84288 4.83784 5.08944 4.6445  4.82259 5.18085 5.
7896746 chr1:564951-565019  10.4398 9.69074 8.42758 9.17384 10.0964 10.3752 10.1012 10.1138 11.3821 10.2215 10.3044 10.4538 9.97446 9.20296 10.275
7896748 chr1:566062-566129  9.86568 9.80962 8.65148 9.67507 10.278  10.3569 10.2154 9.52325 10.9151 10.0506 10.0778 10.3406 10.4719 8.9431  10.375
7896750 chr1:568069-568136  4.8142  6.82916 5.08015 6.31573 6.72305 7.16969 6.51835 5.24391 4.83771 6.81832 5.84991 6.63447 7.3016  5.55058 6.5767
7896752 M37726  10.8945 10.1632 9.02309 10.0032 10.7147 10.4226 10.7769 9.96899 10.824  10.5344 10.7493 10.4421 10.4779 9.52137 10.5517 10.0235 10
7896754 LOC100287934    7.62758 6.96861 6.836   6.42229 6.30081 6.15117 6.3138  7.29051 6.55117 6.4928  6.97607 6.16855 6.13747 7.07129 6.40922 6.
7896756 FAM87A  5.23801 4.9829  5.12344 5.52534 5.05098 5.0566  4.78944 4.91503 4.5412  5.01875 4.56597 4.65835 4.57686 4.78197 5.24035 4.97885 5.
7896759 LINC01128   6.87835 7.03432 6.92744 6.51797 5.95858 5.22107 7.08375 7.23024 6.70166 5.8114  6.98013 6.49939 5.91376 7.16258 6.66227 7.3413
7896761 SAMD11  6.49093 6.28924 6.08082 6.24777 6.65234 6.07944 6.63608 6.01455 6.50566 6.62279 5.52158 6.35826 6.63139 5.89787 6.63591 6.15299 6.
7896779 KLHL17  6.5201  6.27045 5.91752 6.25505 6.59764 6.4753  6.63558 6.14231 6.39466 6.64137 5.76425 6.34283 6.48685 5.94846 6.63098 6.23207 6.
```

Find what: #

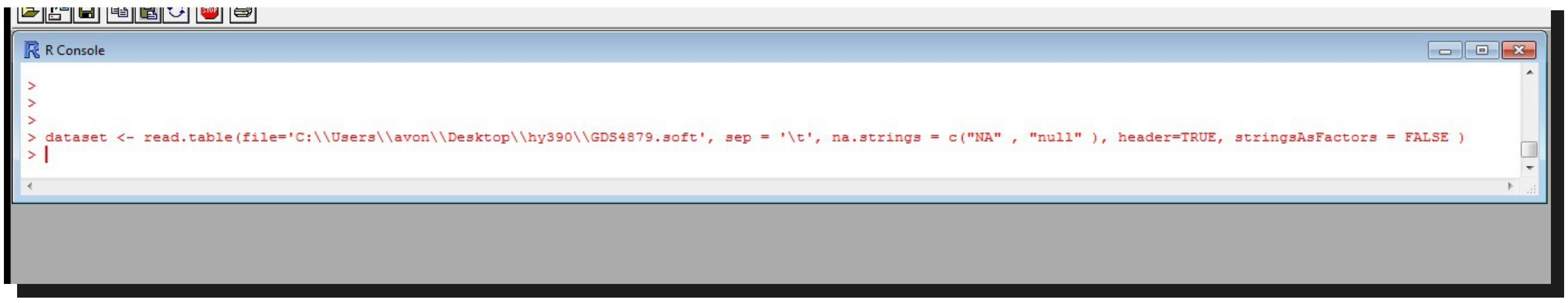
[✓] Match case

Search: Current document

[60:2] : 33386   Modified   ANSI   LF   INS   1 character(s) selected.

3. Read in the dataset in R.
Seperate with tab character '\t'
Set the NA string (na.strings) as either "NA" or "null"

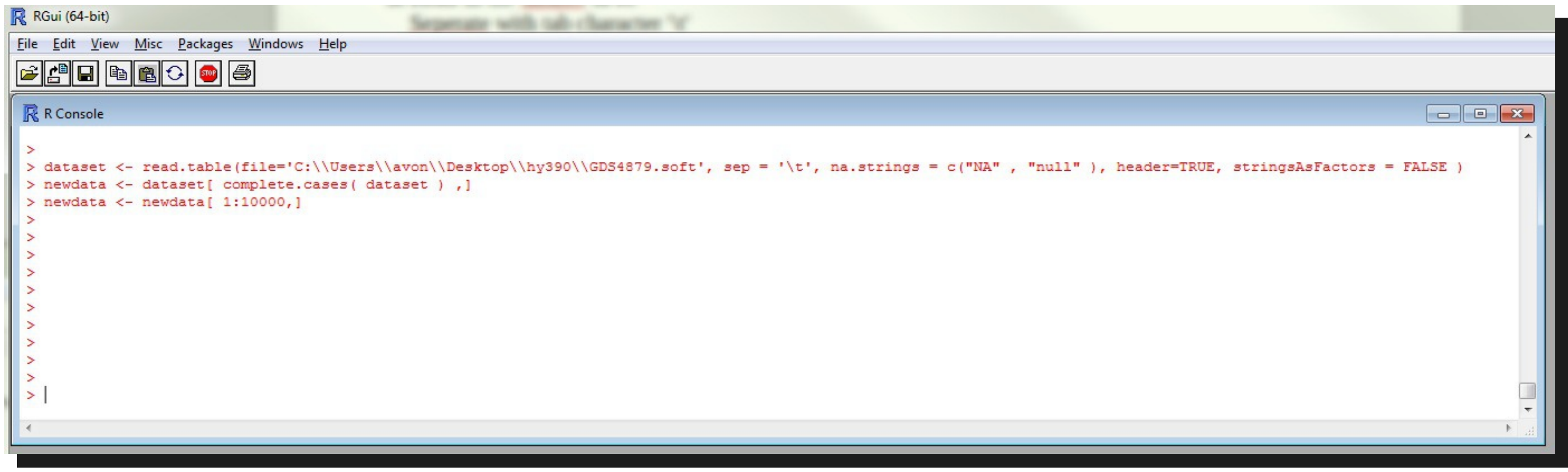
```
dataset <-read.table
(file='C:\\Users\\avon\\Desktop\\hy390\\GDS4879.soft',  sep = '\t',
na.strings = c("NA" , "null" ), header=TRUE,
stringsAsFactors = FALSE)
```



```
>
>
>
> dataset <- read.table(file='C:\\Users\\avon\\Desktop\\hy390\\GDS4879.soft', sep = '\t', na.strings = c("NA" , "null" ), header=TRUE, stringsAsFactors = FALSE )
> 
```

4. Choose only a subset of lines for the final dataset : all lines from 1 - 10000
that contain no "NA" or "null".
Remove all NA's with complete cases and then take the first 10000 lines.

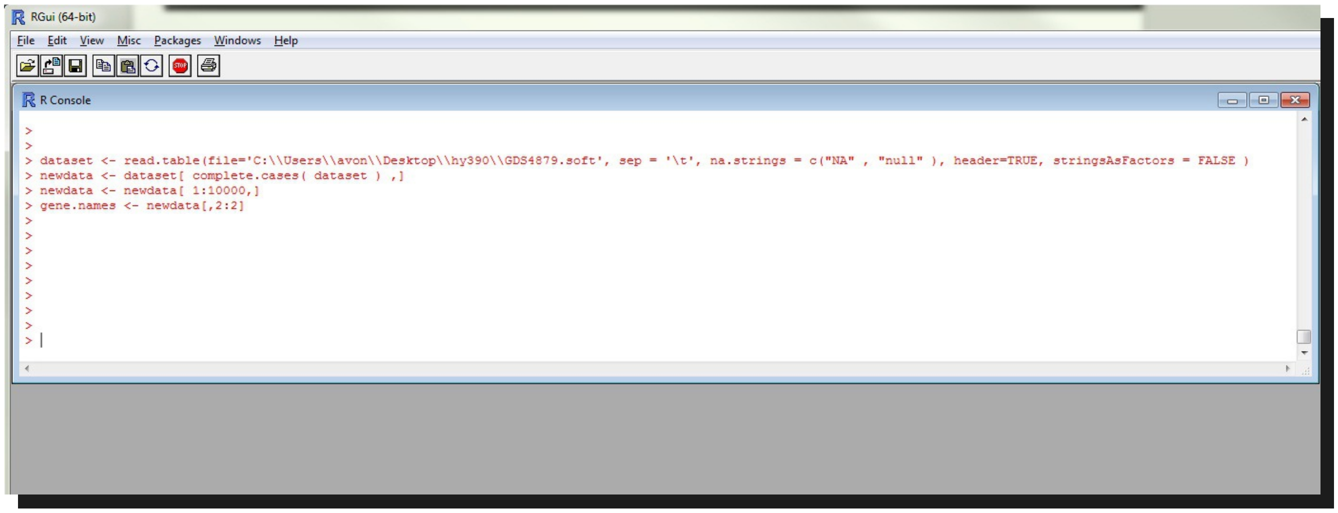
```
newdata <- dataset[ complete.cases( dataset ) ,]
newdata <- newdata[ 1:10000,]
```



```
>
> dataset <- read.table(file='C:\\Users\\avon\\Desktop\\hy390\\GDS4879.soft', sep = '\t', na.strings = c("NA" , "null" ), header=TRUE, stringsAsFactors = FALSE )
> newdata <- dataset[ complete.cases( dataset ) ,]
> newdata <- newdata[ 1:10000,]
>
>
>
>
>
>
>
>
>
>
> |
```

5. Save the second column that contains information about the gene names
in a variable called gene.names .

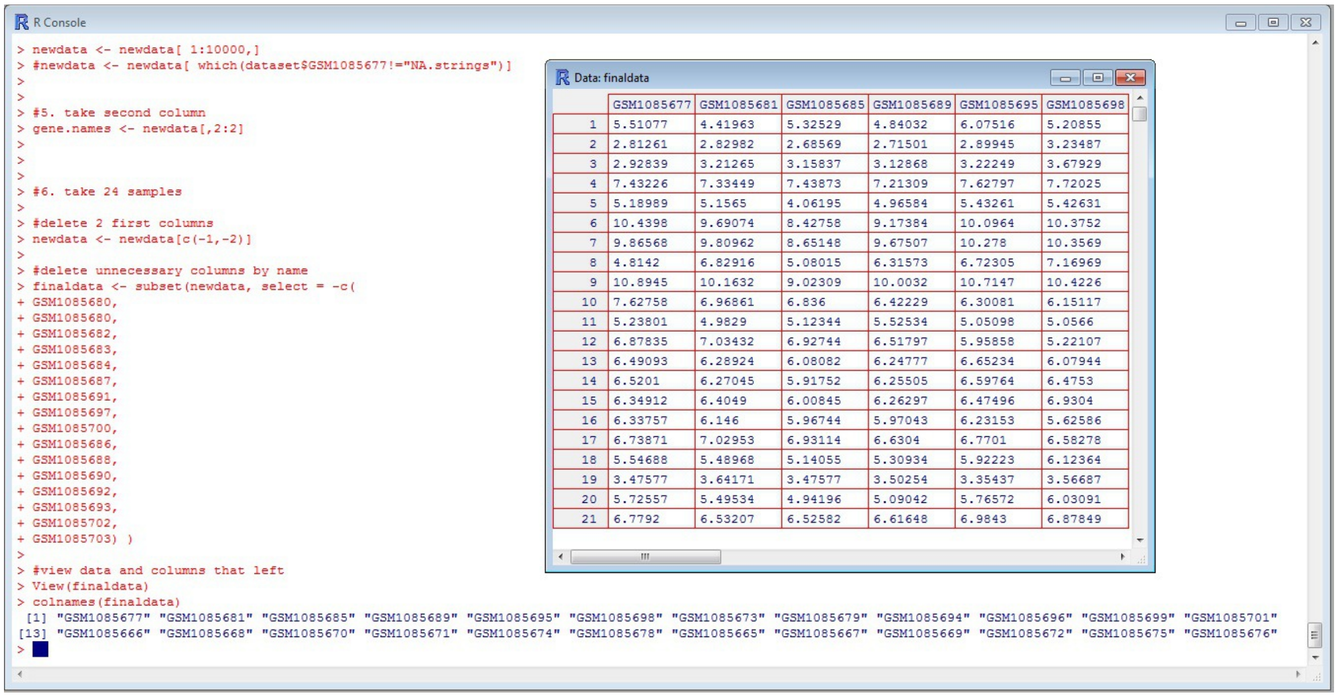
```
gene.names <- newdata[,2:2]
```



```
>
>
> dataset <- read.table(file='C:\\Users\\avon\\Desktop\\hy390\\GDS4879.soft', sep = '\t', na.strings = c("NA" , "null" ), header=TRUE, stringsAsFactors = FALSE )
> newdata <- dataset[ complete.cases( dataset ) ,]
> newdata <- newdata[ 1:10000,]
> gene.names <- newdata[,2:2]
>
>
>
>
>
>
>
>
>
> |
```

6. Take The final dataset that consist only of a subset of 24 sample that we need

```
#delete 2 first columns
newdata <- newdata[c(-1,-2)]

#delete unnecessary columns by name
finaldata <- subset(newdata, select =
-c(GSM1085680,GSM1085680,GSM1085682,GSM1085683,GSM1085684,GSM1085687,
GSM1085691,GSM1085697,GSM1085700,GSM1085686,GSM1085688,GSM1085690,GSM
1085692,GSM1085693,GSM1085702,GSM1085703) )

#view data and columns that left
View(finaldata)
colnames(finaldata)
```



```
> newdata <- newdata[ 1:10000,]
> #newdata <- newdata[ which(dataset$GSM1085677!="NA.strings")]
>
>
> #5. take second column
> gene.names <- newdata[,2:2]
>
>
>
> #6. take 24 samples
>
> #delete 2 first columns
> newdata <- newdata[c(-1,-2)]
>
> #delete unnecessary columns by name
> finaldata <- subset(newdata, select = -c(
+ GSM1085680,
+ GSM1085680,
+ GSM1085682,
+ GSM1085683,
+ GSM1085684,
+ GSM1085687,
+ GSM1085691,
+ GSM1085697,
+ GSM1085700,
+ GSM1085686,
+ GSM1085688,
+ GSM1085690,
+ GSM1085692,
+ GSM1085693,
+ GSM1085702,
+ GSM1085703) )
>
> #view data and columns that left
> View(finaldata)
> colnames(finaldata)
 [1] "GSM1085677" "GSM1085681" "GSM1085685" "GSM1085689" "GSM1085695" "GSM1085698" "GSM1085673" "GSM1085679" "GSM1085694" "GSM1085696" "GSM1085699" "GSM1085701"
[13] "GSM1085666" "GSM1085668" "GSM1085670" "GSM1085671" "GSM1085674" "GSM1085678" "GSM1085665" "GSM1085667" "GSM1085669" "GSM1085672" "GSM1085675" "GSM1085676"
>
```

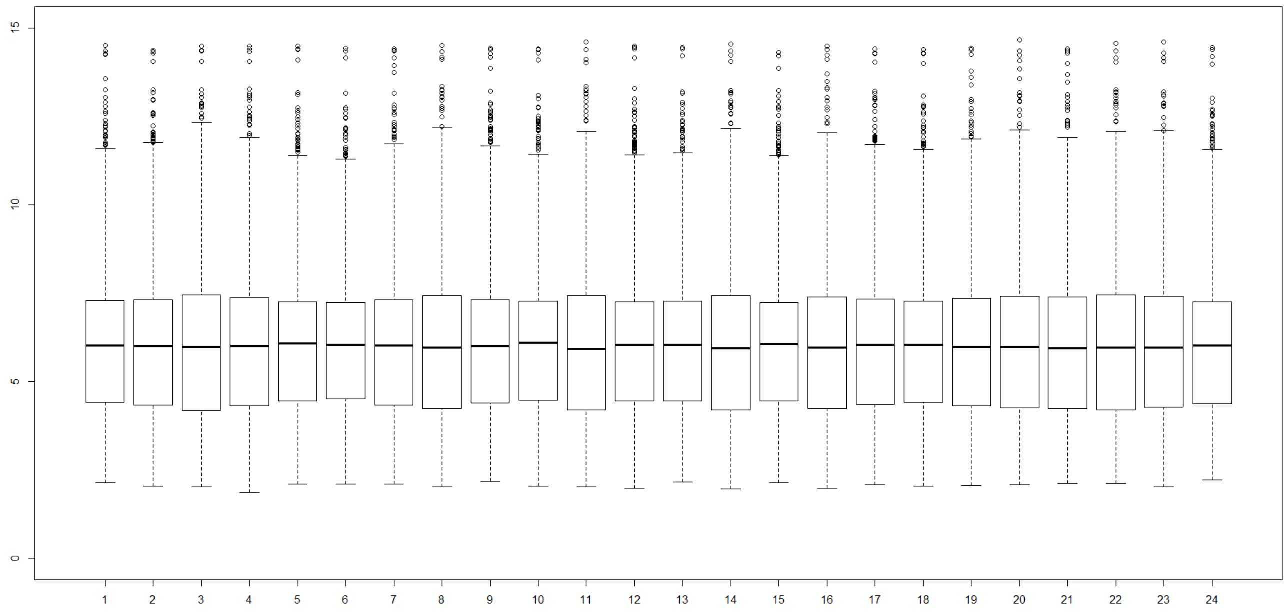
Data: finaldata

	GSM1085677	GSM1085681	GSM1085685	GSM1085689	GSM1085695	GSM1085698
1	5.51077	4.41963	5.32529	4.84032	6.07516	5.20855
2	2.81261	2.82982	2.68569	2.71501	2.89945	3.23487
3	2.92839	3.21265	3.15837	3.12868	3.22249	3.67929
4	7.43226	7.33449	7.43873	7.21309	7.62797	7.72025
5	5.18989	5.1565	4.06195	4.96584	5.43261	5.42631
6	10.4398	9.69074	8.42758	9.17384	10.0964	10.3752
7	9.86568	9.80962	8.65148	9.67507	10.278	10.3569
8	4.8142	6.82916	5.08015	6.31573	6.72305	7.16969
9	10.8945	10.1632	9.02309	10.0032	10.7147	10.4226
10	7.62758	6.96861	6.836	6.42229	6.30081	6.15117
11	5.23801	4.9829	5.12344	5.52534	5.05098	5.0566
12	6.87835	7.03432	6.92744	6.51797	5.95858	5.22107
13	6.49093	6.28924	6.08082	6.24777	6.65234	6.07944
14	6.5201	6.27045	5.91752	6.25505	6.59764	6.4753
15	6.34912	6.4049	6.00845	6.26297	6.47496	6.9304
16	6.33757	6.146	5.96744	5.97043	6.23153	5.62586
17	6.73871	7.02953	6.93114	6.6304	6.7701	6.58278
18	5.54688	5.48968	5.14055	5.30934	5.92223	6.12364
19	3.47577	3.64171	3.47577	3.50254	3.35437	3.56687
20	5.72557	5.49534	4.94196	5.09042	5.76572	6.03091
21	6.7792	6.53207	6.52582	6.61648	6.9843	6.87849

Task 2

1. Make a boxplot of the samples
First convert finaldata to numeric.

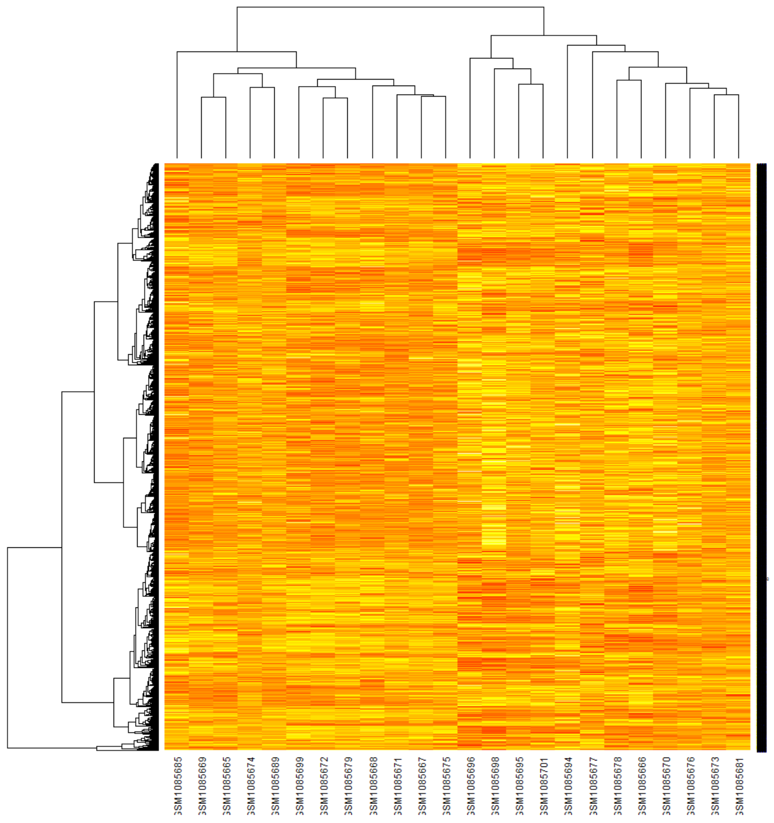
```
dataNum <- matrix(data = NA, nrow = dim(finaldata)[1], ncol =
dim(finaldata)[2])

for (i in 1:dim(finaldata)[2])
{
    dataNum[,i] <- c(as.numeric(finaldata[[i]]))
}

boxplot(dataNum , ylim= c(0,15))
```



As shown above the dataset is normalized because one or more arrays don't have intensity levels which are drastically different from the rest of the arrays, this may indicate a problem with these arrays. As normalization refers to the creation of shifted and scaled versions of statistics, where the intention is that these normalized values allow the comparison of corresponding normalized values for different datasets in a way that eliminates the effects of certain gross influences, as in an anomaly time series.

Task 3
1. Construct a heatmap of all samples

```
#---create table
nba <- finaldata
matrix <- data.matrix(nba)

#---create heatmap
heatmap <- heatmap(matrix)
```



```
Heatmap reflexes gene expression values . Heatmap has a graphical
representation of data , where the individual values of each sample
contained in a matrix , are represented as colors
```

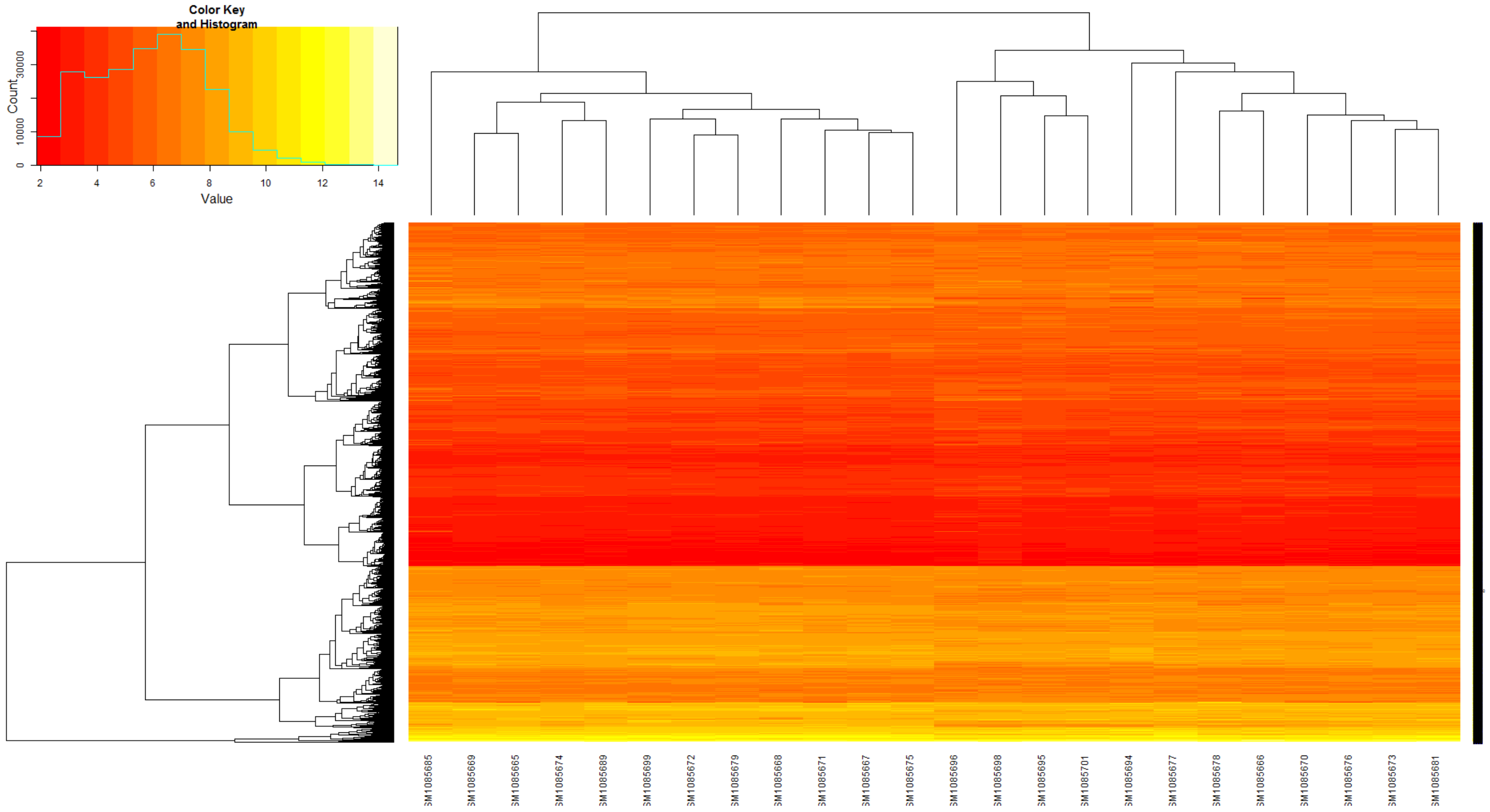
2. Construct a heatmap of all samples using the function heatmap.2 from the library gplots. Use scale = 'none' and trace = 'none'.

First Install gplots library :

```
if (!require("gplots")) {
   install.packages("gplots", dependencies = TRUE)
   library(gplots)
   }
```

Then create 2nd heatmap using the function heatmap.2 :

```
heatmap.2 <- heatmap.2(matrix, scale="none",  trace = "none")
```

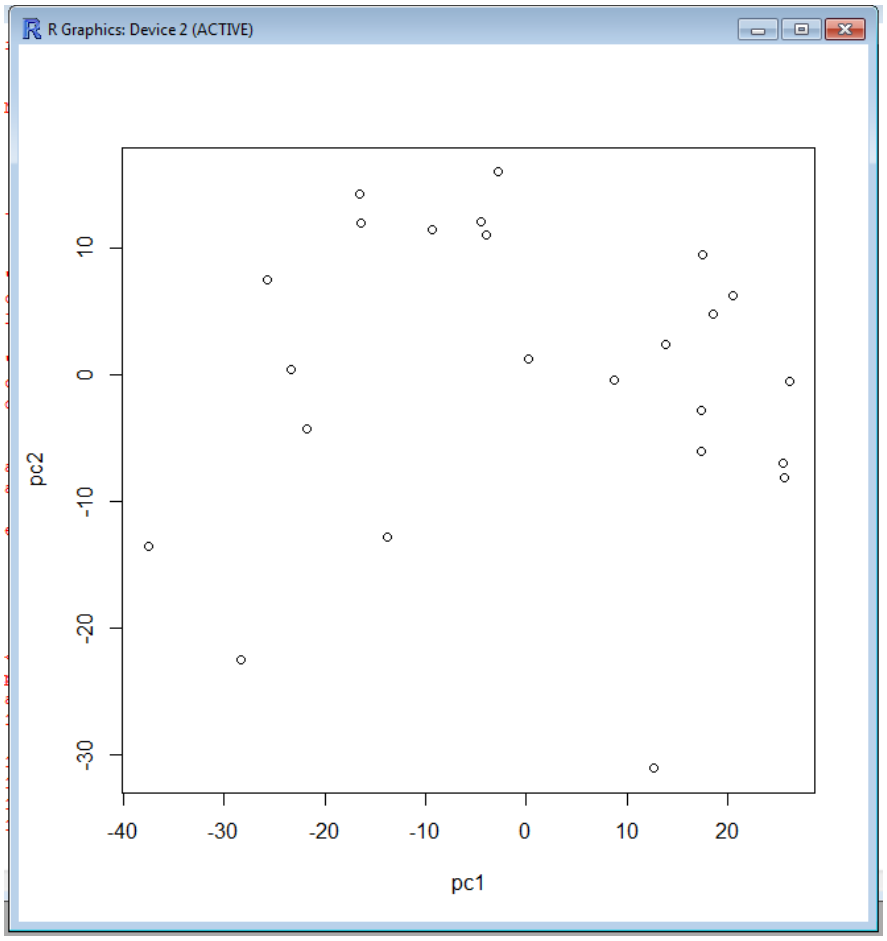


The differences betwween Heatmap with method 1 heatmap with method 2 is that 2nd heatmap's rows and columns are reordered according to some set of values within the restrictions imposed by the dendrogram is carried out.
2Nd heatmap has also a histogram added to the top left side.

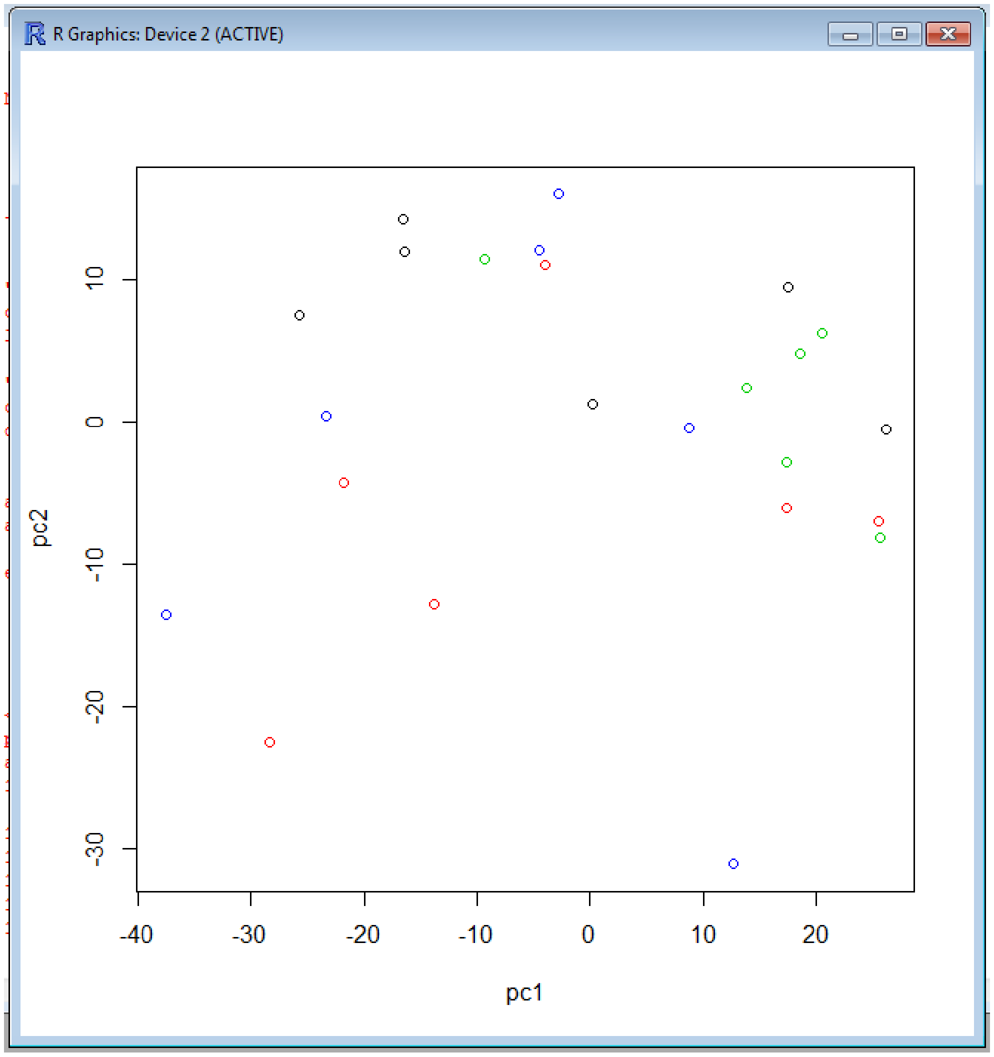
Task 4

```
finaldatanew <- t(matrix)

temp <- prcomp(finaldatanew)
colors <-as.factor
(c(rep('yellow',6),rep('green',6),rep('blue',6),rep('red',6)))

#plotno1
plot(temp$x[,1], temp$x[,2], xlab = 'pc1', ylab = 'pc2')

#plotno2
plot(temp$x[,1], temp$x[,2], xlab = 'pc1', ylab = 'pc2',col=colors)
```

4. Plot the data using the Principal Component 1 and Principal Component 2.

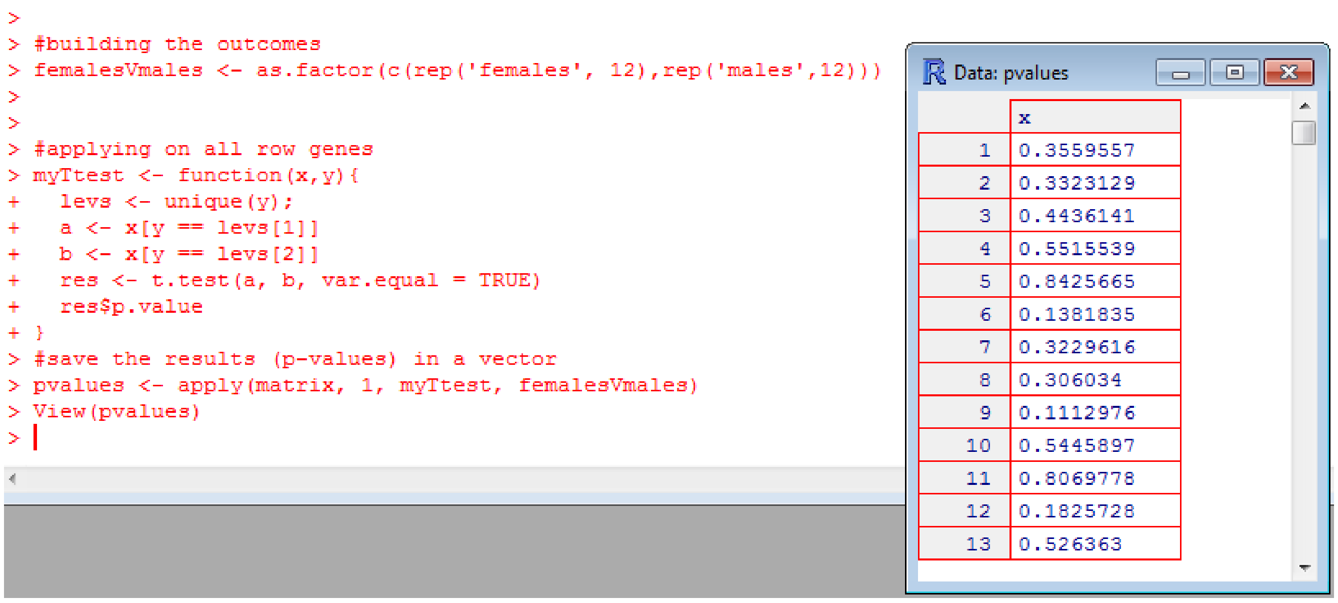


5. Try to color the data points of the plot according to the class of each sample. e.g. use red for female-alcoholic, green for female non-alcoholic, etc.



Task 5

```
#building the outcomes
femalesVmales <- as.factor(c(rep('females', 12),rep('males',12)))

#applying on all row genes
myTtest <- function(x,y){
  levs <- unique(y);
  a <- x[y == levs[1]]
  b <- x[y == levs[2]]
  res <- t.test(a, b, var.equal = TRUE)
  res$p.value
}

#save the results p-values in a vector
pvalues <- apply(matrix, 1, myTtest, femalesVmales)
View(pvalues)
```



```
>
> #building the outcomes
> femalesVmales <- as.factor(c(rep('females', 12),rep('males',12)))
>
>
> #applying on all row genes
> myTtest <- function(x,y){
+   levs <- unique(y);
+   a <- x[y == levs[1]]
+   b <- x[y == levs[2]]
+   res <- t.test(a, b, var.equal = TRUE)
+   res$p.value
+ }
> #save the results (p-values) in a vector
> pvalues <- apply(matrix, 1, myTtest, femalesVmales)
> View(pvalues)
>
```

	x
1	0.3559557
2	0.3323129
3	0.4436141
4	0.5515539
5	0.8425665
6	0.1381835
7	0.3229616
8	0.306034
9	0.1112976
10	0.5445897
11	0.8069778
12	0.1825728
13	0.526363

```
#2.  ascending order of pvalues_femaleVmale
ordered_femaleVmale <-
pvalues_femaleVmale[order(pvalues_femaleVmale)]

#take the first 100 minimum pvalues
list1=head(ordered_femaleVmale ,100)


for (i in 1:length(list1) ) {
     for (y in 1:length(pvalues_femaleVmale) ) {
          if( pvalues_femaleVmale[y] == list1[i])
               write.table (matrix[y] ,
     file='C:\\Users\\avon\\Desktop\\hy390\\tmp.txt' ,
               append=TRUE,sep = "\t")
     }
}
```

Task 6

```
#1.  applying on all row genes
myTtest <- function(x,y){
  levs <- unique(y);
  a <- x[y == levs[1]]
  b <- x[y == levs[2]]
  res <- t.test(a, b, var.equal = TRUE)
  res$p.value
}

#2.  save the results (p-values) of nonalcoholics and alcoholics in a
vector
pvalues_nonalcoholicsValcoholics <- apply(matrix, 1, myTtest,
nonalcoholicsValcoholics)
#View(pvalues_nonalcoholicsValcoholics)
```

```
#ascending order of pvalues_nonalcoholicsValcoholic
ordered_nonalcoholicsValcoholics <-
pvalues_nonalcoholicsValcoholics[order(pvalues_nonalcoholicsValcoholi
cs)]

#take the first 100 minimum pvalues
list2=head(ordered_nonalcoholicsValcoholics ,100)

for (i in 1:length(list2) ) {
     for (y in 1:length(pvalues_nonalcoholicsValcoholics) ) {
          if( pvalues_nonalcoholicsValcoholics[y] == list2[i])


write.table(matrix[y],file='C:\\Users\\avon\\Desktop\\hy390\\tmp2.txt
',append=TRUE,sep = "\t")
     }
}
```