

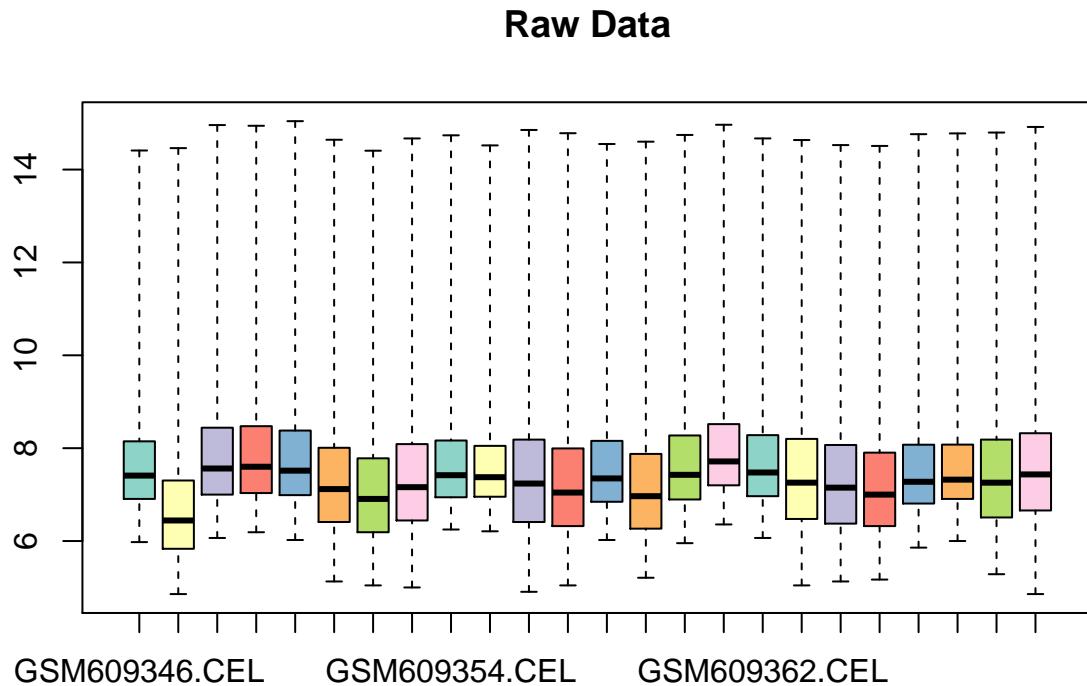
# Lab 1

2226768Y

18 March 2016

Read in the CEL files in the directory, then normalize the data

```
data.raw <- ReadAffy()  
boxplot(data.raw, main= "Raw Data" , col=brewer.pal(8,"Set3") )
```



Function to plot boxplots of expression sets using ggplot

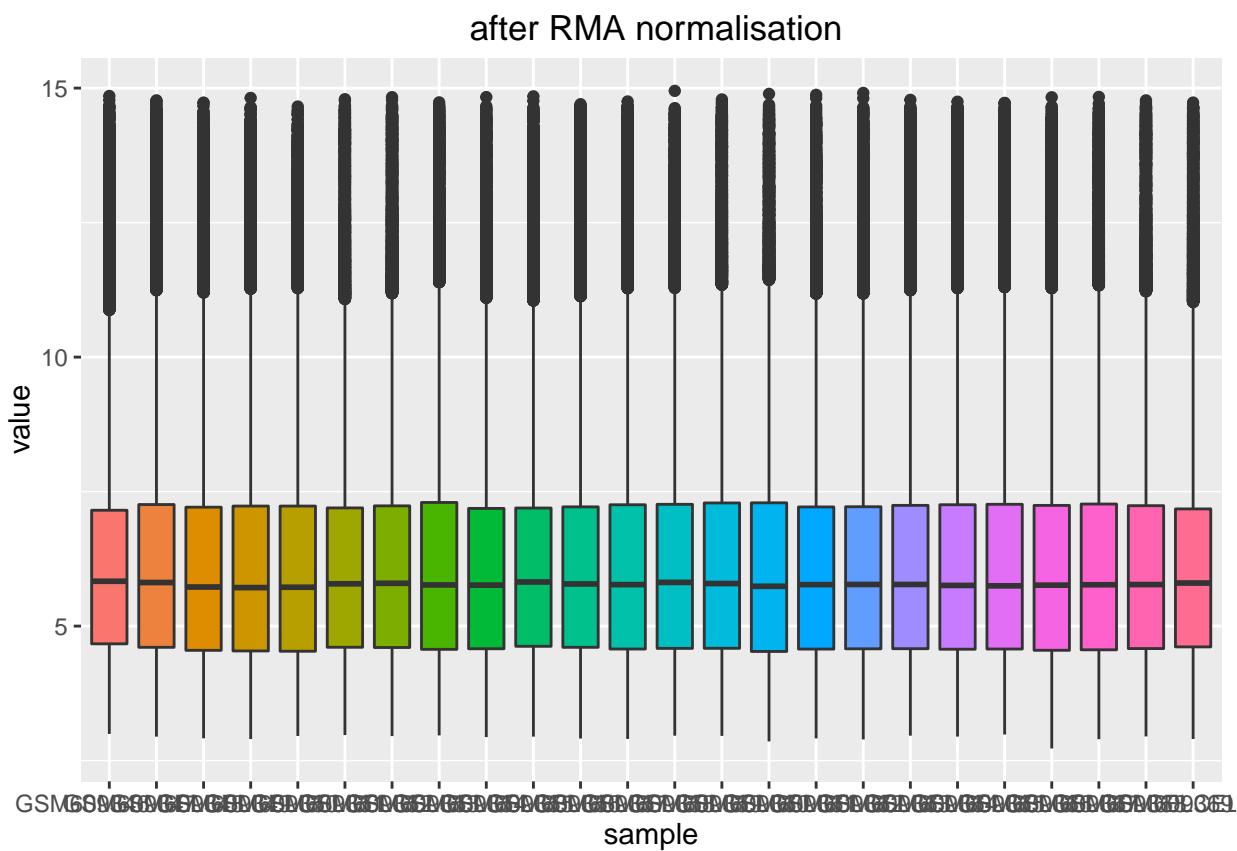
```
ggboxplot <-function ( dat, title){  
  library(reshape2)  
  plotData <- as.data.frame(t(exprs(dat)))  
  plotData$sample <- row.names(plotData)  
  plotData.m <- melt(plotData, id.vars = "sample")  
  
  g<- ggplot(plotData.m, aes(x=sample,y=value, fill= sample )) +  
    geom_boxplot() +guides(fill=FALSE)+ ggttitle(title)  
  return(g)  
}
```

```
eset <- rma(data.raw)
```

Normalisation methods 1) RMA 2) Quantile only

```
## Background correcting  
## Normalizing  
## Calculating Expression
```

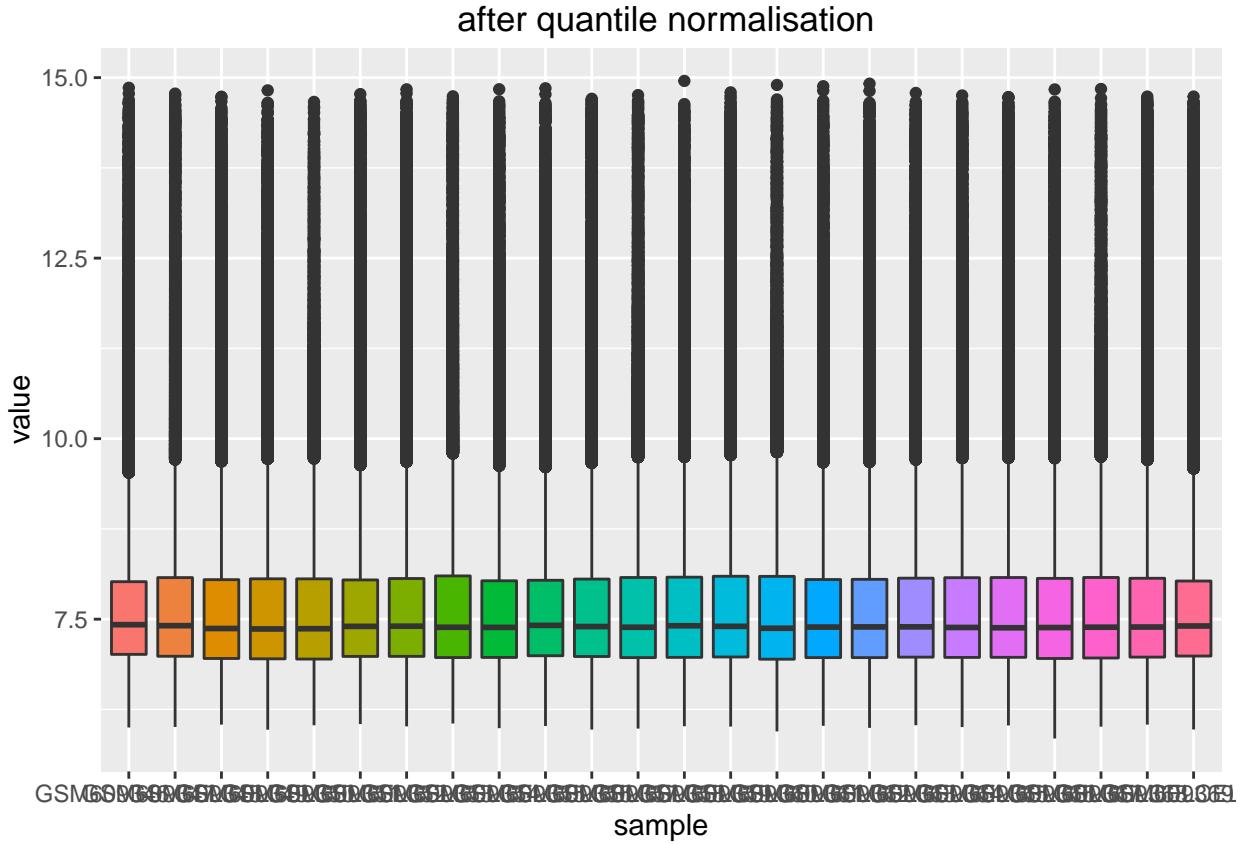
```
ggbboxplot(eset, " after RMA normalisation")
```



```
eset.quantile <- expresso(data.raw, bgcorrect.method="none", normalize.method="quantiles",  
                           pmcorrect.method="pmonly", summary.method="medianpolish")
```

```
## background correction: none  
## normalization: quantiles  
## PM/MM correction : pmonly  
## expression values: medianpolish  
## background correcting...done.  
## normalizing...done.  
## 54675 ids to be processed  
## |  
## |#####|
```

```
ggboxplot (eset.quantile, " after quantile normalisation")
```



**Explore Expression Sets** pData return the pheno data , nothing in there at the moment. sampleNames returns the names of the samples featureNames returns the probe IDs exprs returns the matrix of the expression values, probes X samples.

### Add phenotype data to eset

Read in the phenotype data

```
pheno <- read.table("E-GEO-24739.sdrf.txt", header = TRUE, sep= "\t",
stringsAsFactors = F, row.names = 1, comment.char = '!')
```

Change sample names to same as data in expression set

```
sNames <- paste0(t(as.data.frame(strsplit(row.names(pheno), " "))),[,1], ".CEL")
row.names(pheno) <- sNames
```

Merge them together by row name

```
pd <- pData(eset)
pdat <- merge(pd, pheno, by=0)
rownames(pdat) <- pdat[, 'Row.names']
pdat <- subset(pdat, select=-c(Row.names))
```

Check they have swapped order and then put pdata back in eset

```
all.equal(rownames(pData(eset)), rownames(pdat))

## [1] TRUE

pData(eset) <- pdat
```

## Explore the Data

Get the phenotype data for the disease state and cell phase

```
diseaseState <- factor(eset@phenoData$Characteristics..disease.state. ,
  labels = c("CML", "N"))

cellPhase <- factor(eset@phenoData$Characteristics..cell.cycle.phase. ,
  levels = c( "G1/S/G2/M", "G0") , labels = c("G1", "G0" ))
```

```
DC <- paste(diseaseState, cellPhase, sep = ".")
DC <- factor(DC, levels = c("CML.G0" , "CML.G1", "N.G0", "N.G1"))
design <- model.matrix(~0+ DC)
colnames(design) <- levels(DC)
```

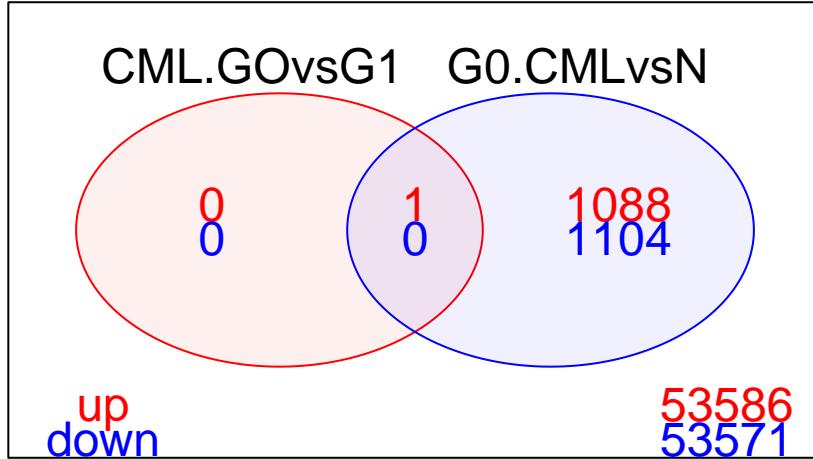
**Design a model matrix to split the data into 4 groups** Make the contrast matrix for comparing 1) the G0 versus G1 in CML cells 2) the G0 phase cells in CMS verses non disease cells

```
con <- c("CML.G0vsG1 =CML.G0-CML.G1", "G0.CMLvsN = CML.G0 -N.G0")
contrast.matrix <- makeContrasts(contrasts = con,levels=design)

fit <- lmFit(eset,design)
fit2 <- contrasts.fit(fit, contrast.matrix)
fit2 <- eBayes(fit2)
```

Plot venn diagram of the number of differentited genes

```
results <- decideTests(fit2)
vennDiagram(results,names = c("CML.G0vsG1", "G0.CMLvsN") , include=c("up", "down"),
  counts.col=c("red", "blue"),circle.col = c("red", "blue", "green3"))
```



Pick for top ten probes with the highest fold-change for the G0 cells CML vs Normal.

```
colnames(coef(fit))

## [1] "CML.G0" "CML.G1" "N.GO"    "N.G1"

T <- topTable(fit2, coef= 2, sort.by = "logFC", number=10)
kable (T, digits= 4)
```

	logFC	AveExpr	t	P.Value	adj.P.Val	B
228766_at	4.4349	6.3399	5.2650	0.0000	0.0041	2.6380
209894_at	4.3677	8.0668	4.0625	0.0005	0.0219	-0.1324
206488_s_at	4.3039	7.4632	5.2641	0.0000	0.0041	2.6360
204304_s_at	-4.2967	8.5106	-3.6108	0.0015	0.0421	-1.1639
209555_s_at	4.2736	7.9655	4.4551	0.0002	0.0126	0.7744
206834_at	4.2088	9.3394	5.2606	0.0000	0.0042	2.6280
204753_s_at	-4.1766	6.7312	-5.8394	0.0000	0.0018	3.9308
213515_x_at	4.1186	6.9887	3.8507	0.0009	0.0295	-0.6185
217388_s_at	4.1157	6.9771	4.4173	0.0002	0.0132	0.6869
205984_at	-3.8826	7.2335	-7.2328	0.0000	0.0003	6.8954

# Labs 2 and 3: PCA and Clustering

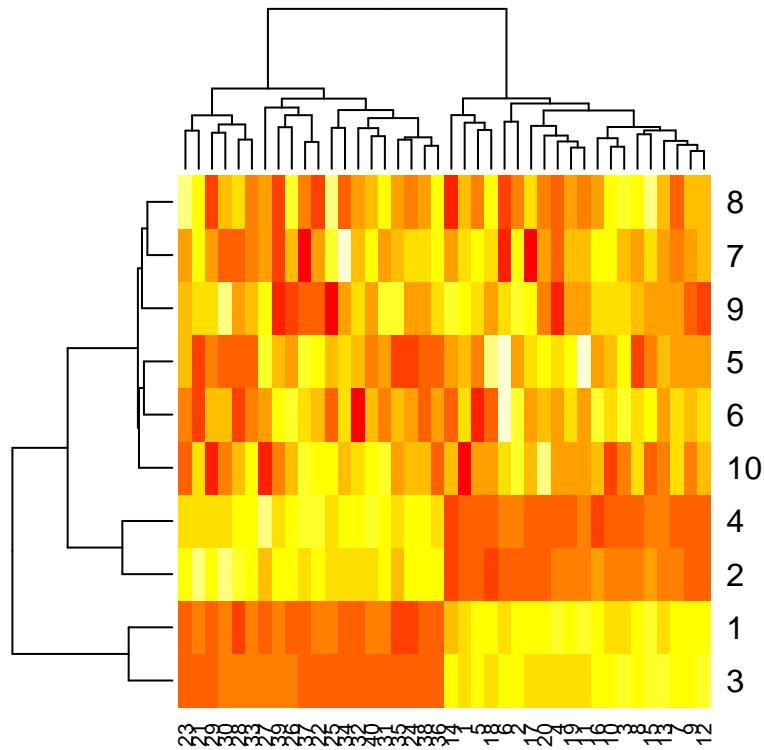
2226768Y

9 March 2016

```
means <-c(3,-3,-2,2,2,-5,-2 ,5)
ma<- matrix(data=NA, 10,40)
set.seed(1)
for (i in 1:4){
  j<- 2*(i-1) +1
  ma[i,1:20] <- mvrnorm(20, means[j],1)
  ma [i,21:40] <- mvrnorm(20, means[j+1] ,1)
}

for (i in 5:10)
  ma [i,] <- rnorm( 40, 4,1)
```

## 1. Generating data Plot a Heatmap

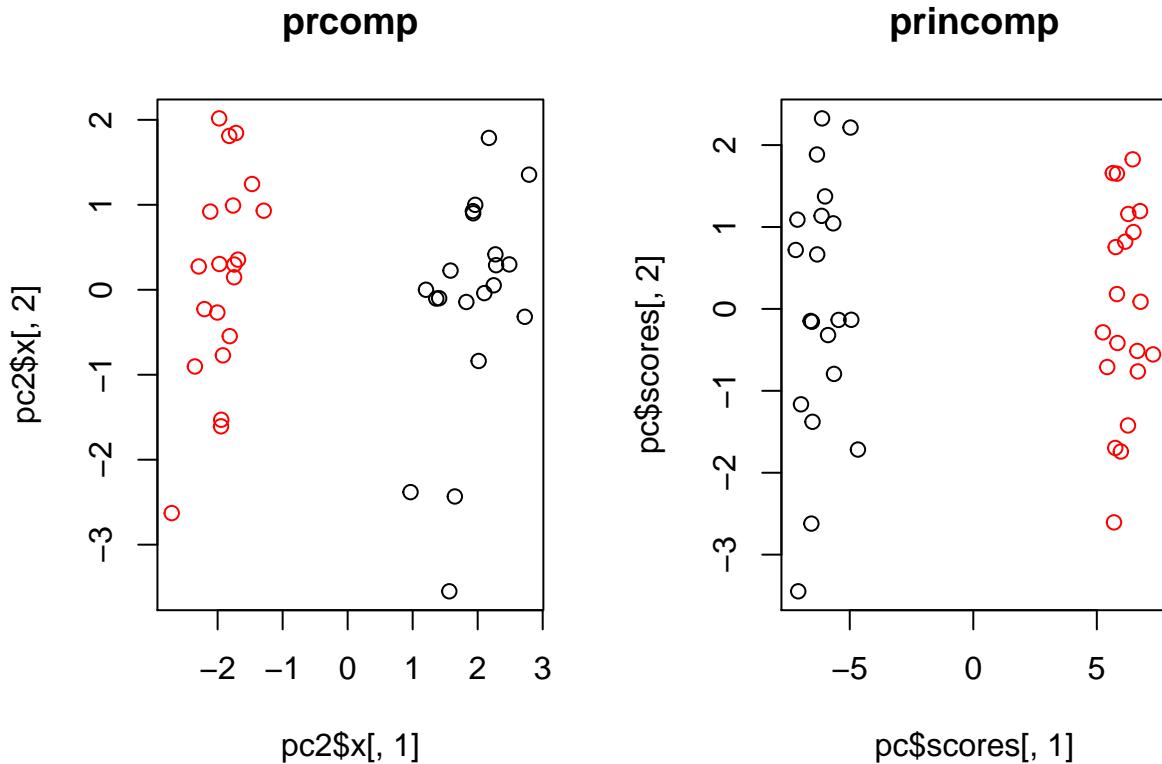


Plot PCA

```

pc <- princomp(t(ma),scale. =T,center = T)
pc2 <- prcomp (t(ma),scale. =T,center = T, retx =T)
cond = c("healthy","disease")
metadata <- rep (cond,each = 20)
par(mfrow = c(1, 2))
plot (pc2$x[,1],pc2$x[,2],col = as.factor(metadata),main="prcomp" )
plot (pc$scores[,1],pc$scores[,2],col = as.factor(metadata),main="princomp")

```



This shows very similar separation. Note scaling difference on PC axis 1.

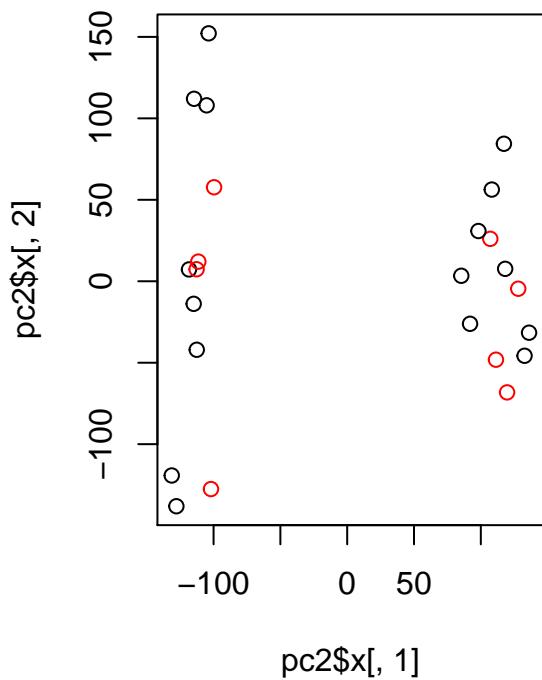
#### 2. PCA on Microarray Data

```

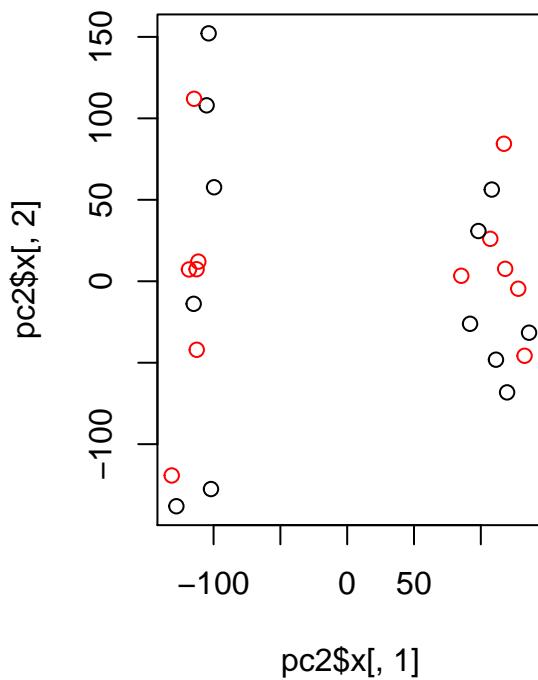
# Transpose data use prcomp and plot results
pc2 <- prcomp (t(exData),scale. =T,center = T, retx =T)
par(mfrow = c(1, 2))
plot (pc2$x[,1],pc2$x[,2],col = phenoType$disease.state,main= "PCA coloured on Disease State" )
plot (pc2$x[,1],pc2$x[,2],col = phenoType$cell.phase,main= "PCA coloured on Cell Phase")

```

**PCA coloured on Disease State**



**PCA coloured on Cell Phase**



The data does not separate on the phenotype of the data meaning the biggest changes in differentiation are not due to cell type or disease state.

#### 4. pick top 10 genes from PC1 and PC2

```
top10PC1<- data.frame(sort(abs(pc2$rotation[,1]),decreasing = TRUE)[1:10])
kable(top10PC1)
```

sort.abs.pc2.rotation...1....decreasing...TRUE..1.10.	
AFFX-DapX-M_at	0.0085324
AFFX-r2-Bs-dap-M_at	0.0085312
AFFX-r2-Bs-dap-3_at	0.0085252
AFFX-DapX-3_at	0.0085151
215450_at	0.0085111
224760_at	0.0084925
213409_s_at	0.0084841
229420_at	0.0084819
207688_s_at	0.0084793
216515_x_at	0.0084782

```
top10PC2<- data.frame(sort(abs(pc2$rotation[,2]),decreasing = TRUE)[1:10])
kable(top10PC2)
```

---

sort.abs.pc2.rotation...2...decreasing...TRUE..1.10.	
228418_at	0.0123467
213984_at	0.0121544
220199_s_at	0.0120594
209852_x_at	0.0119917
225926_at	0.0119758
203428_s_at	0.0118907
227108_at	0.0117744
222714_s_at	0.0117568
223077_at	0.0117036
228578_at	0.0116812

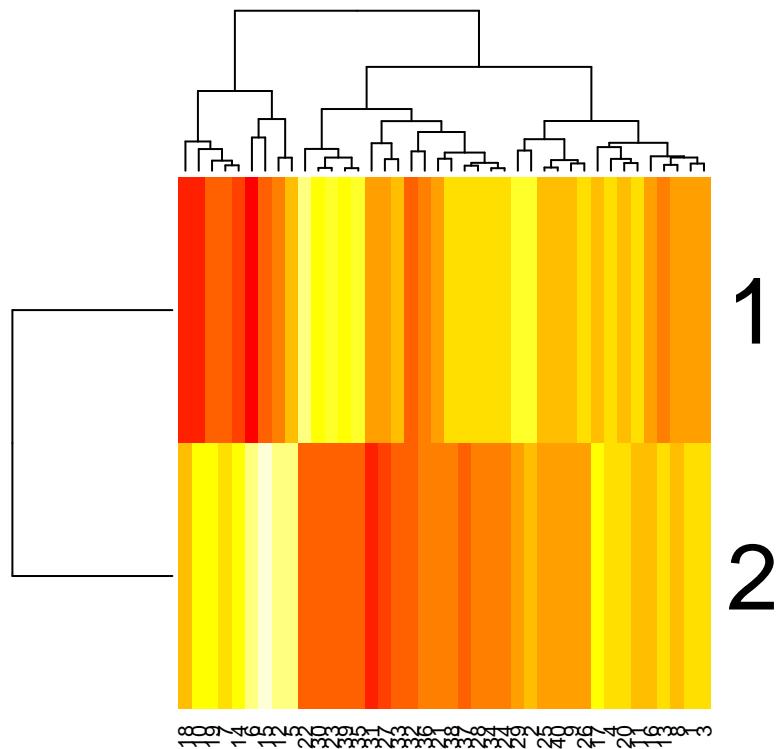
---

# Lab 3

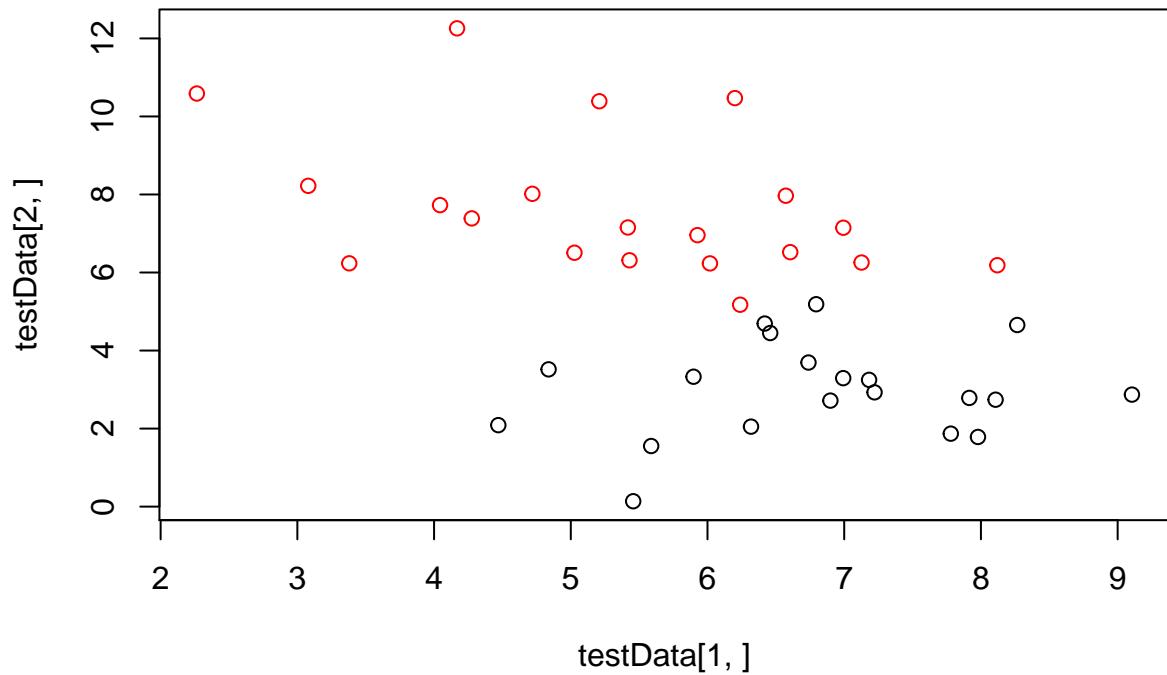
## 1. Hierarchical Clustering on Test Data

Generate a test data set in 2.D ie 2 by n 40 matrix

```
meansNormal <- rnorm(20,3.8,2)
meansDisease <- rnorm (20,4.3,1.5)
testData<- matrix(data=NA, 2,40)
for (i in 1:2){
  testData[i,1:20] <- mvrnorm(20, meansNormal[i],1.5)
  testData[i,21:40] <- mvrnorm(20, meansDisease[i] ,1.5)
}
heatmap(testData)
```



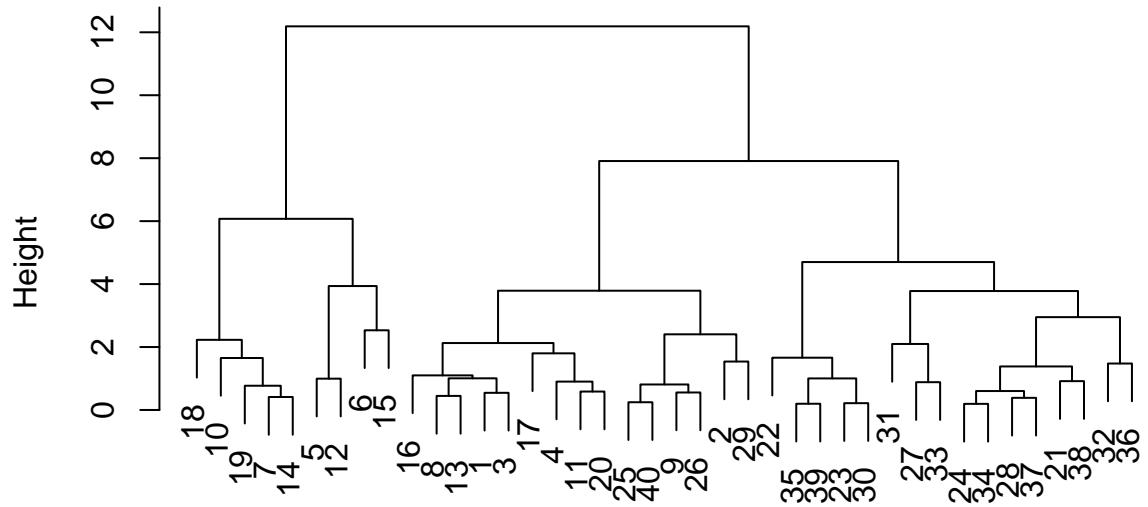
```
plot (testData[1,],testData[2,],col = as.factor(metadata) )
```



2) Hierarchical Clustering

```
# Complete linkage
d = dist(t(testData))
hc = hclust(d,method="complete",members=NULL)
plot(hc)
```

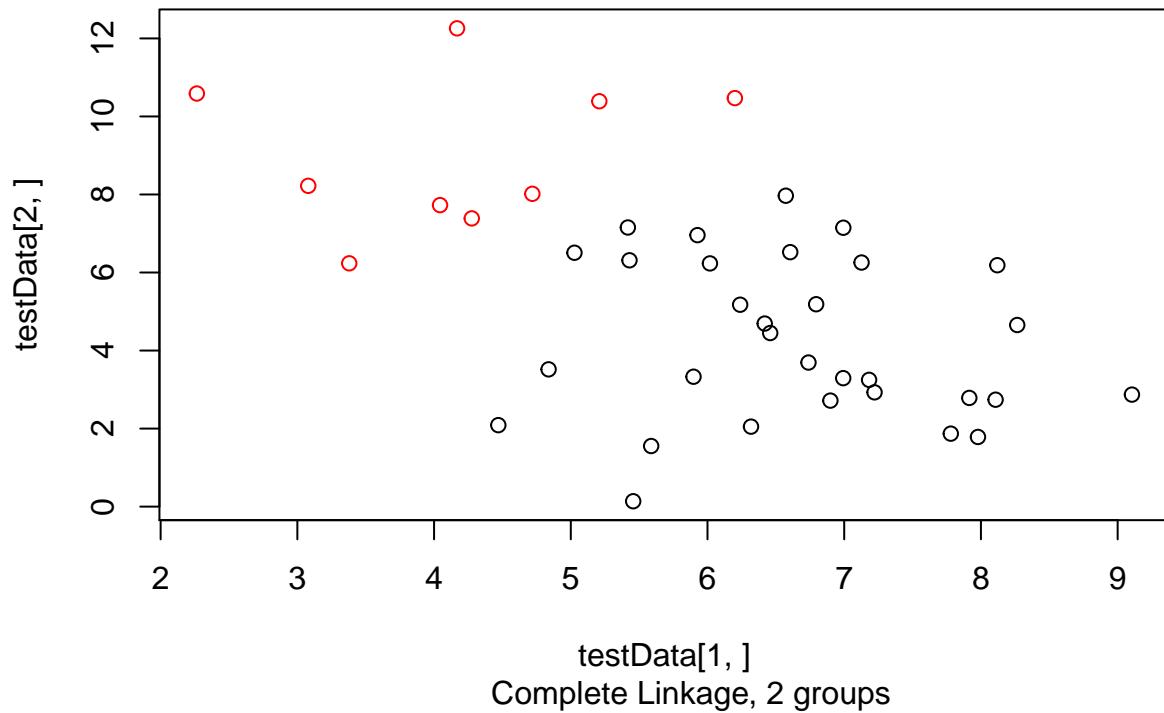
## Cluster Dendrogram



d  
hclust (\*, "complete")

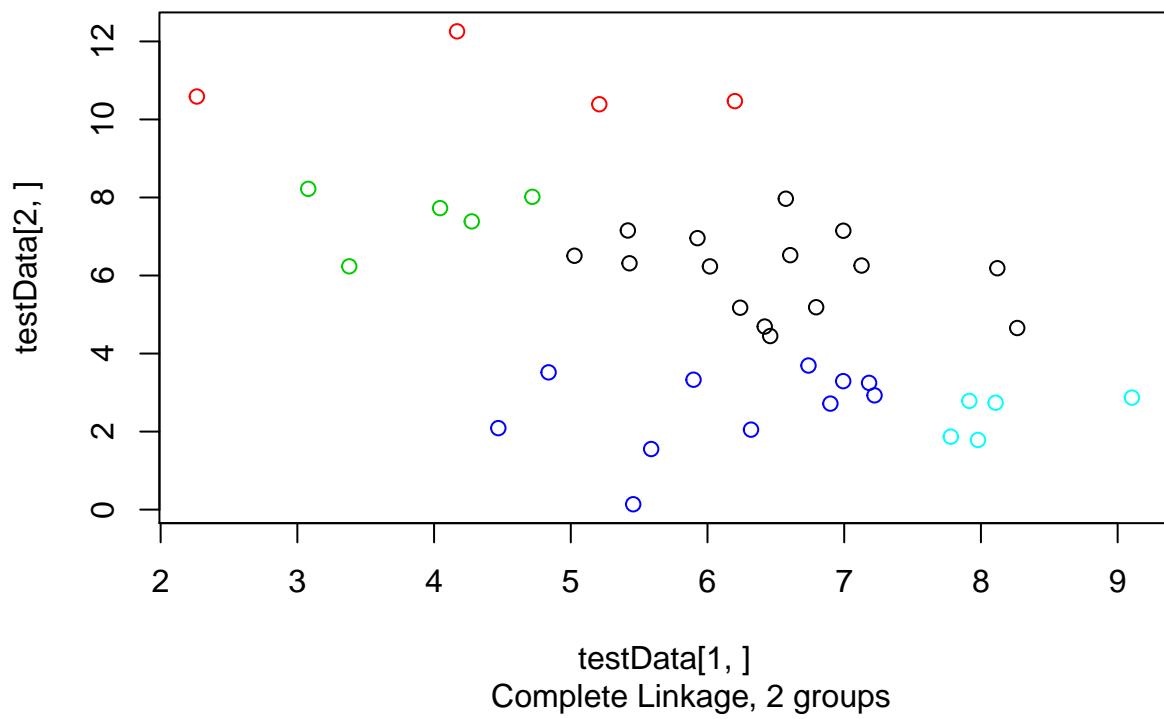
```
ct1 <- cutree (hc , k=2)
plot (testData[1,], testData[2,], col = ct1, main = "Hierarchical clustering" ,sub ="Complete Linkage, 2 groups" )
```

## Hierarchical clustering



```
ct2 <- cutree(hc,5)
plot (testData[1,],testData[2,],col = ct2, main = "Hierarchical clustering" ,sub ="Complete Linkage, 2 groups" )
```

## Hierarchical clustering

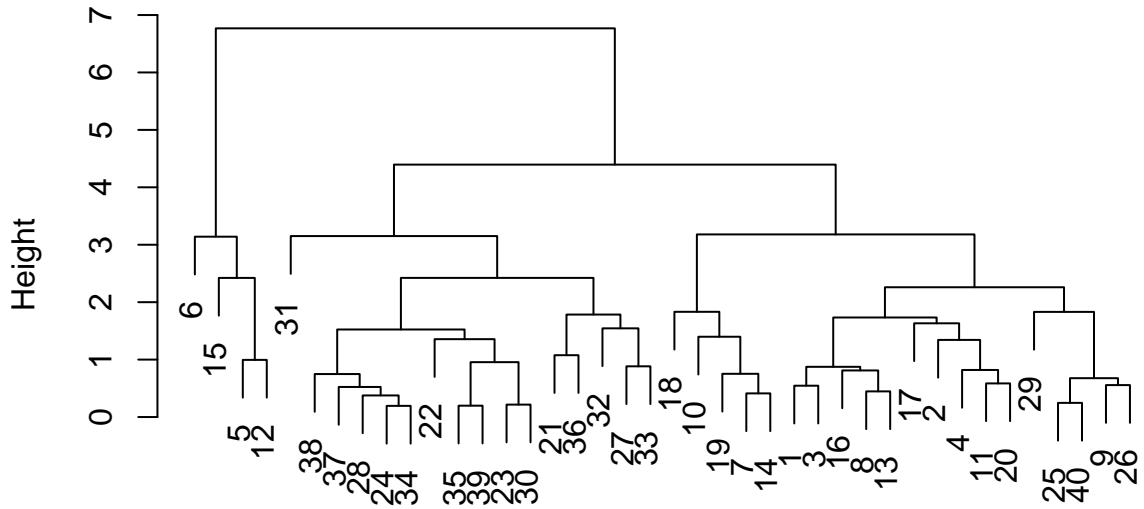


testData[1, ]  
Complete Linkage, 2 groups

Average Linkage

```
hca = hclust(d,method="average",members=NULL)
plot(hca,main = "Hierarchical clustering , average linkage" )
```

## Hierarchical clustering , average linkage

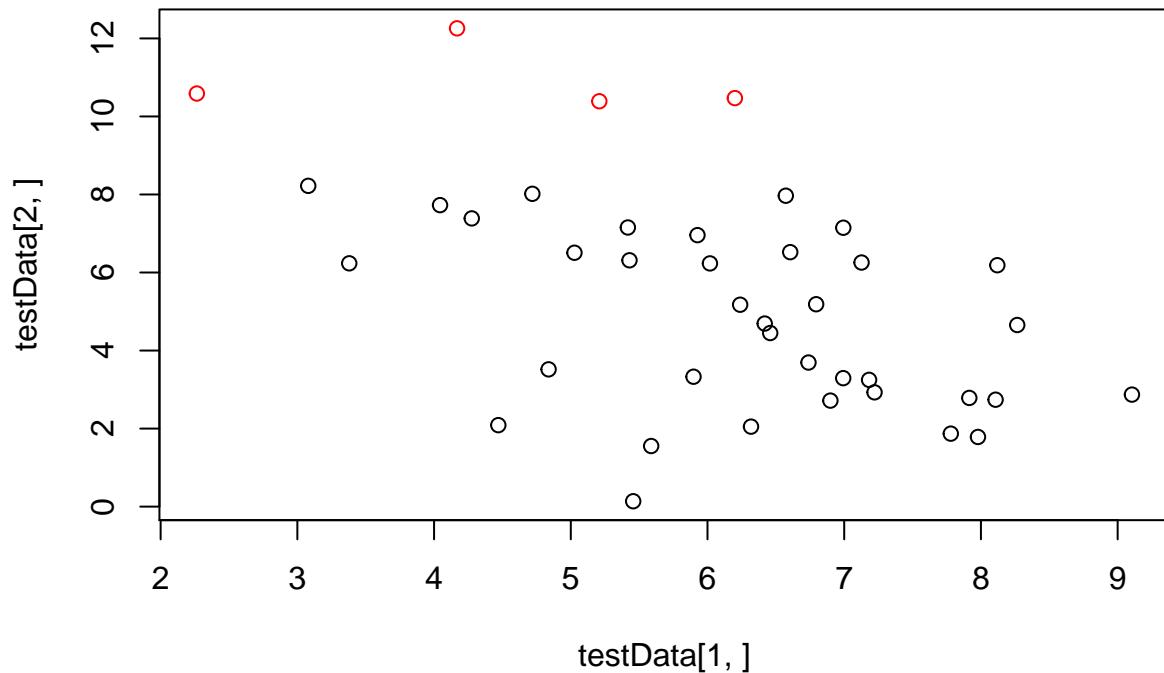


d  
hclust (\*, "average")

```
ct <- cutree (hca , k=2)

plot (testData[1], testData[2], col = ct, main = "Hierarchical clustering , average linkage, 2 groups")
```

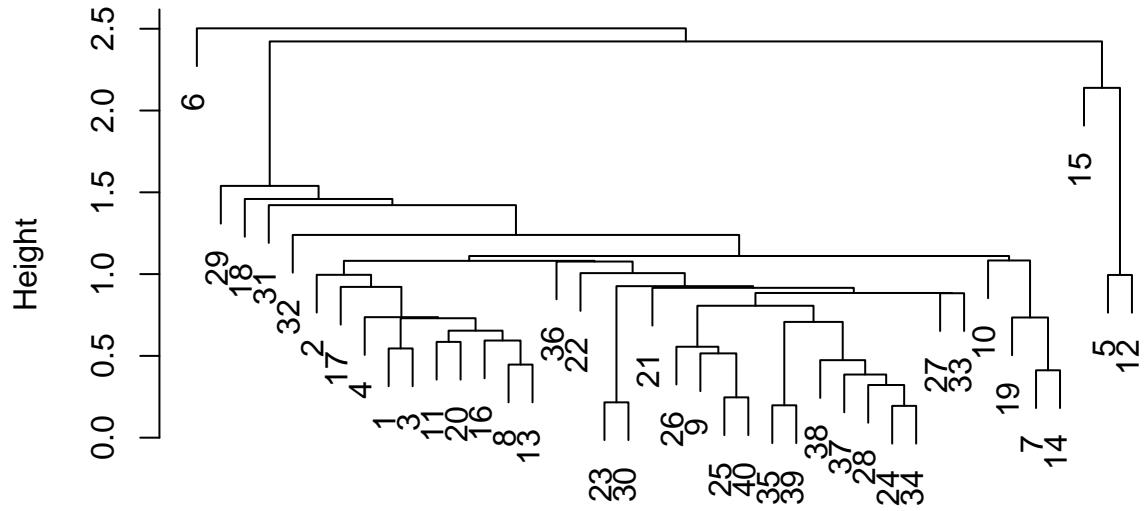
## Hierarchical clustering , average linkage, 2 groups



Single Linkage

```
hcs = hclust(d,method="single",members=NULL)
plot(hcs)
```

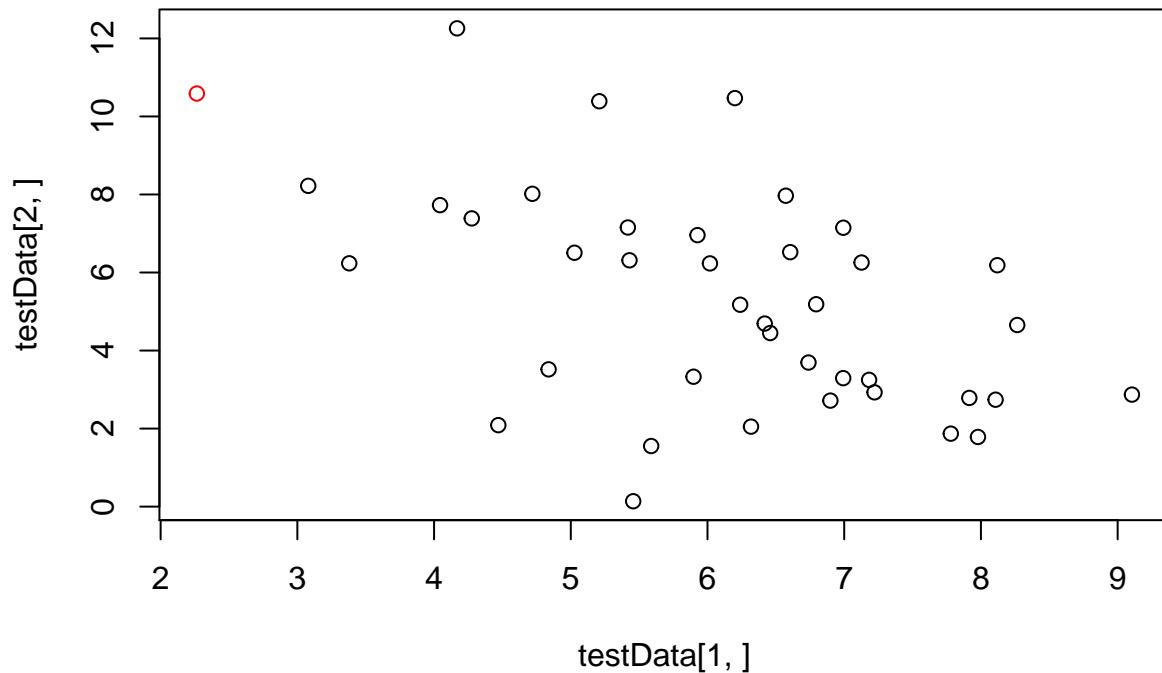
## Cluster Dendrogram



d  
hclust (\*, "single")

```
ct <- cutree (hcs , k=2)
plot (testData[1,],testData[2,],col = ct, main = "Hierarchical clustering , single linkage, 2 groups" )
```

## Hierarchical clustering , single linkage, 2 groups



### 5) Using Heirarchechal Clustering on Real Data

For array data need to append H or D onto front of sample name so can use labelCol

```
dis <- factor(phenoType$disease.state, labels= c("D", "H"))
cell <- factor(phenoType$cell.phase , labels= c("G", "C"))

lab <- 1
for (i in 1:length(cell)){
  lab[i] <-  paste( dis[i],cell[i],i,sep = "_")
}
```

Function to colour and label the nodes

```
colorCodes <- c(H = "red", D = "blue",G ="red",C ="blue")
#function to set label color
labelCol <- function(x) {
  if (is.leaf(x)) {
    ## fetch label
    label <- attr(x, "label")
    code <- substr(label, 1, 1)
    code
    ## use the following line to reset the label to one letter code
    # attr(x, "label") <- code
  }
}
```

```

        attr(x, "nodePar") <- list(lab.col=colorCodes[code])
    }
    return(x)
}

```

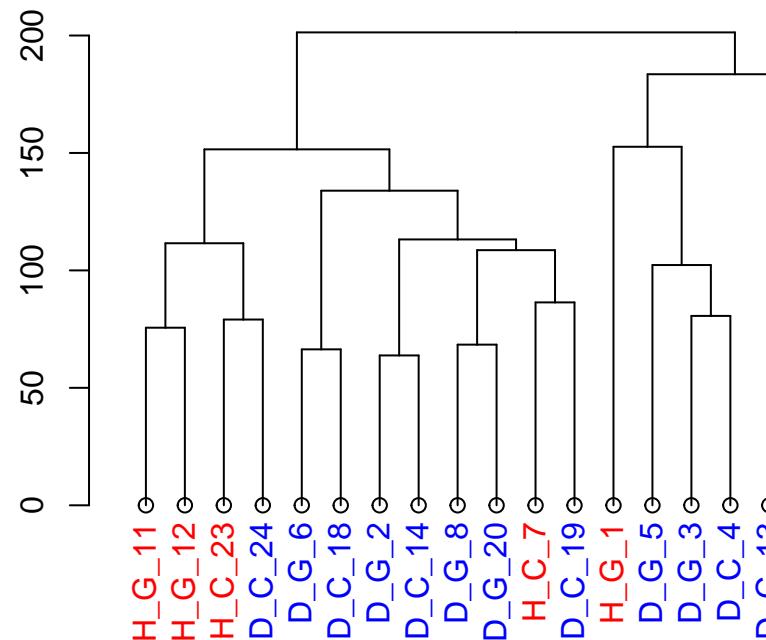
```

exDatSxG <- t(exData)
d = dist((exDatSxG))

hcma = hclust(d,method="complete",members=NULL )
hcma$labels= lab
#plot(hcma, main= "CML Data , Complete Linkage")
ct <- cutree (hcma , k=10)
den <- dendrapply(as.dendrogram(hcma) , labelCol)
plot(den, main = "Complete Linkage")

```

**Complete Linkage**



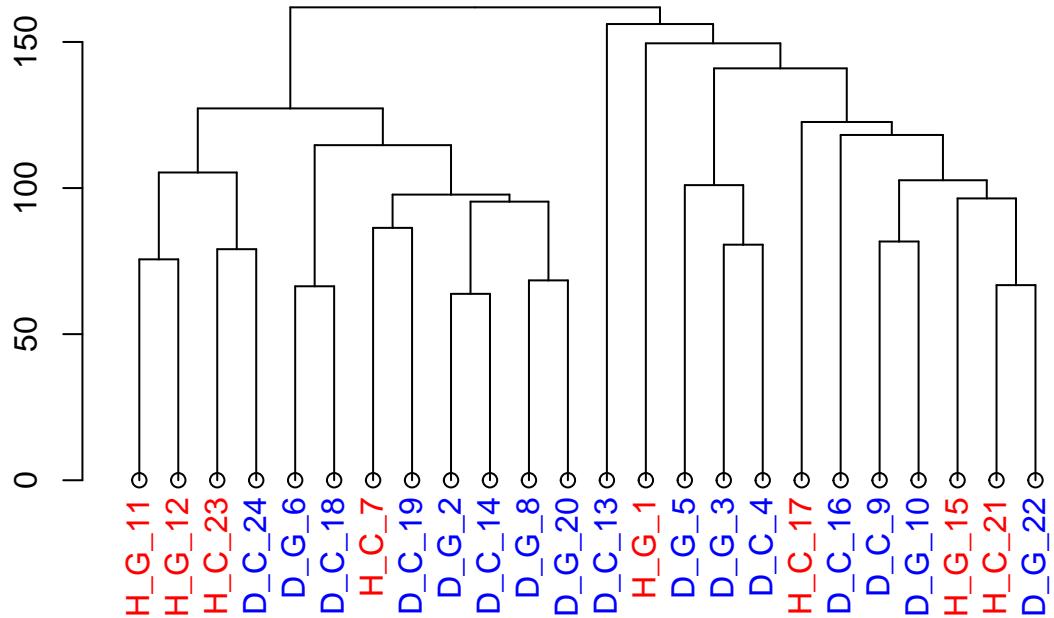
Different Clustering methods for CML Data

```

hca = hclust(d,method="average",members=NULL)
hca$labels <- lab
ct <- cutree (hca , k=20)
den <- dendrapply(as.dendrogram(hca),labelCol)
plot(den, main = "Average Linkage")

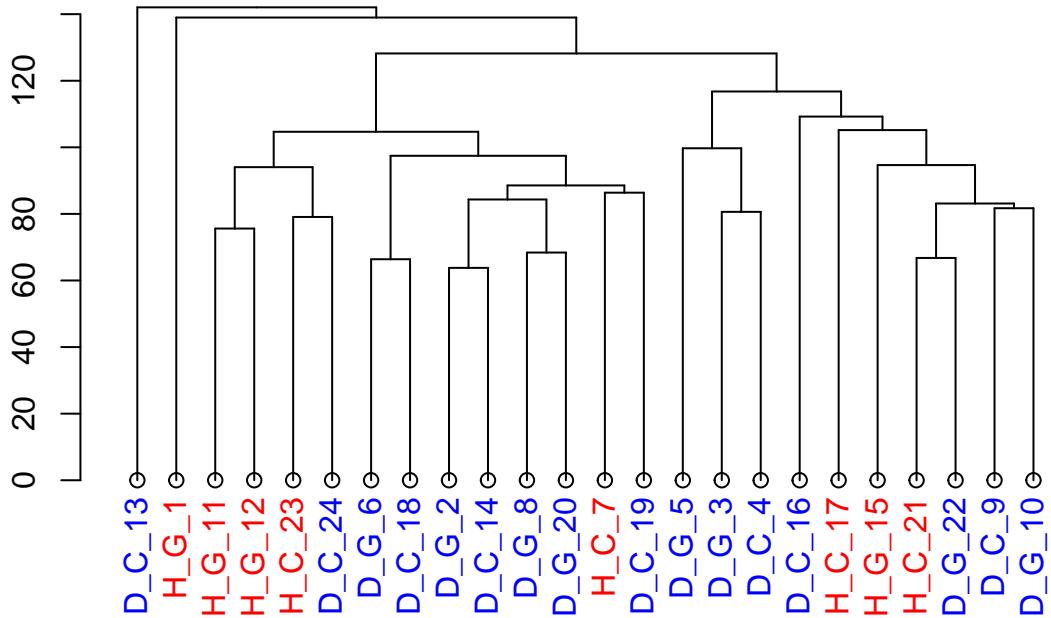
```

## Average Linkage



```
hcs = hclust(d,method="single",members=NULL)
hcs$labels <- lab
den <- dendrapply(as.dendrogram(hcs),labelCol)
plot(den, main = "Single Linkage")
```

## Single Linkage



**Q6) Clustering Genes** Selectig a random subset of genes cluster to avoid computer crashes.

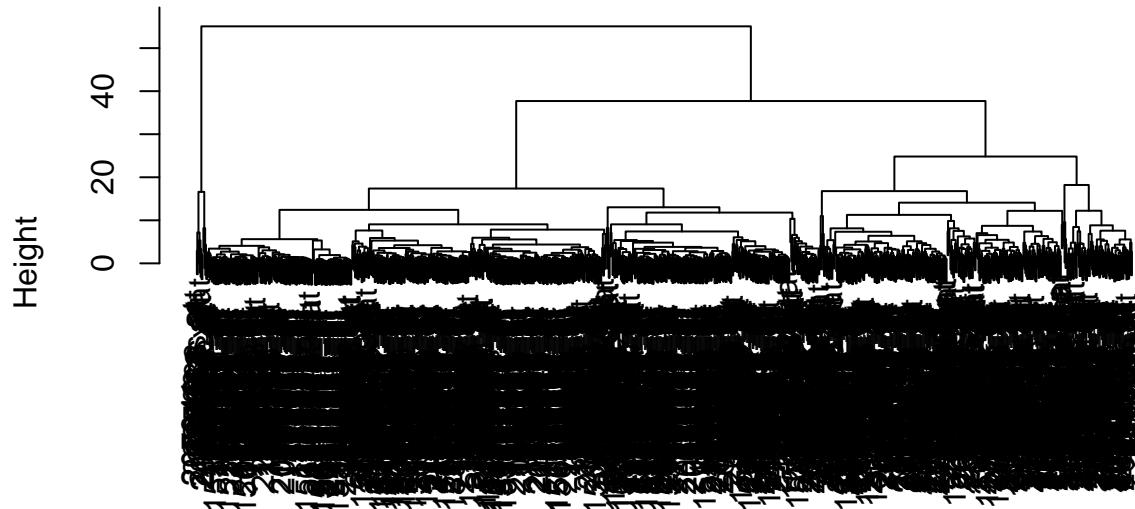
```
set.seed(100)
genes1000 <- sample (rownames(exData),1000)
data.genes1000 <- exData[ genes1000,]
```

Use these in clustering. First have to make array of these gene.

```
distg <- dist(data.genes1000)
hcma = hclust(distg,method="complete",members=NULL )

plot(hcma, main= "CML Data, Complete Linkage")
```

## CML Data, Complete Linkage

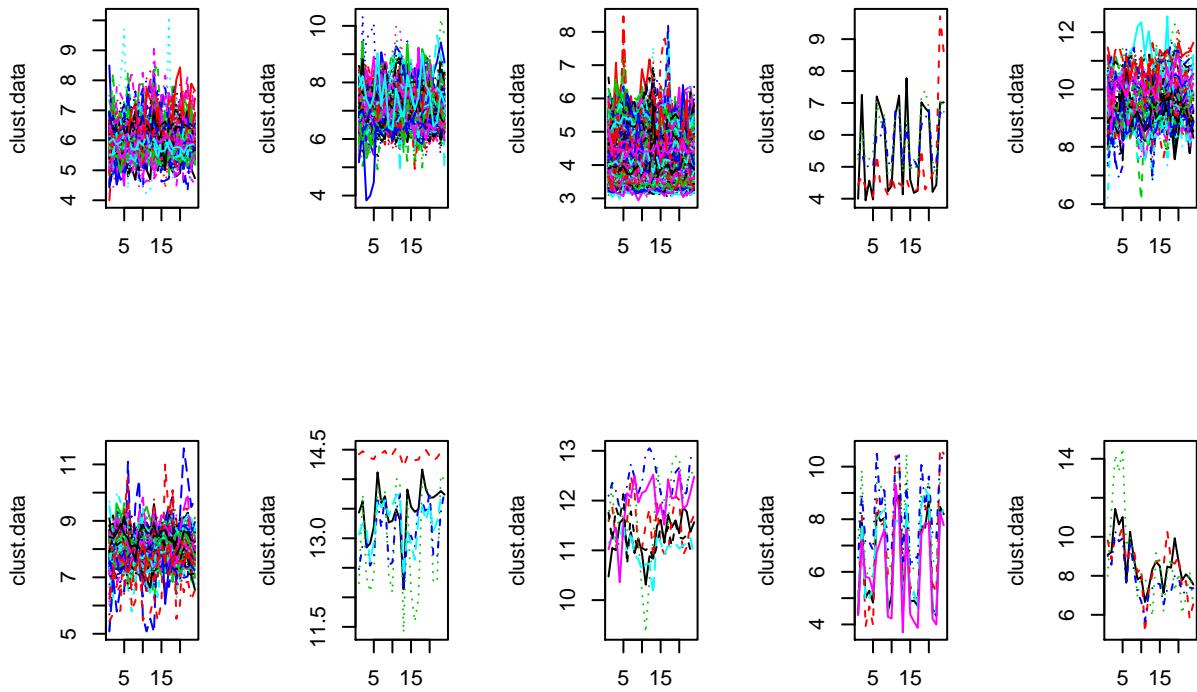


```
distg  
hclust (*, "complete")
```

```
ct <-cutree (hcma , k=10)
```

Plot all the clusters.

```
par(mfrow = c(2, 5))  
  
for (i in 1:10){  
  clustNames <- genes1000[as.vector(ct==i)]  
  clust.data <- t(exData[clustNames,])  
  matplot(clust.data, type = "l")  
}
```



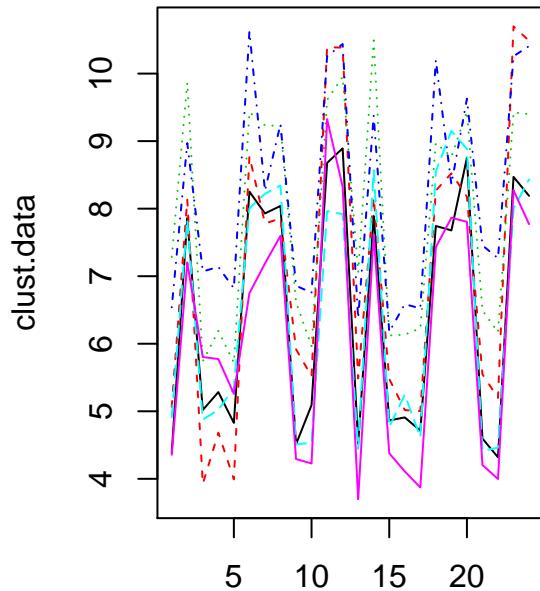
These are randomly selected genes , but some of the smaller clusters show a correlation between the genes. Run a differant linkage and pick the most visually correlated cluster.

```
#Plot 8th cluster
par(mfrow = c(1, 2))
clust8Names <- genes1000[as.vector(ct==9)]
clust.data <- t(exData[clust8Names,])
matplot(clust.data, type = "l", main= "Complete linkage, C9")

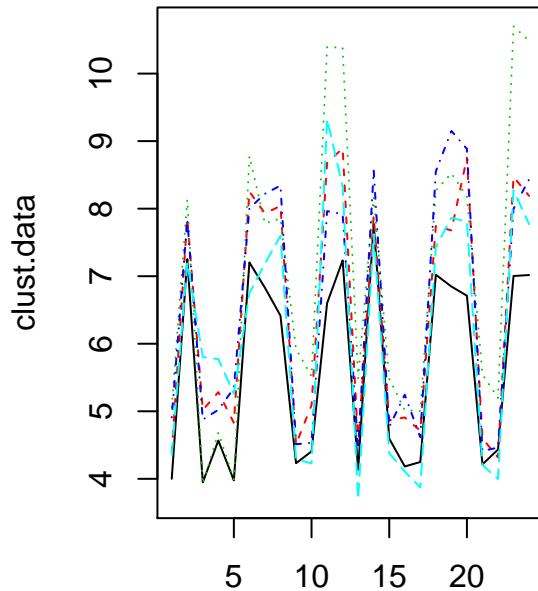
# Average Linkage
hca = hclust(distg,method="average",members=NULL)

ctAv <-cutree (hca , k=10)
clust2Names <- genes1000[as.vector(ctAv==4)]
clust.data <- t(exData[clust2Names,])
matplot(clust.data, type = "l",main="Average linkage, C6")
```

**Complete linkage, C9**



**Average linkage, C6**



Both clusters look very similar.

```
length(clust2Names)  
  
## [1] 5  
  
length(clust8Names)  
  
## [1] 6  
  
sum(clust2Names %in% clust8Names)  
  
## [1] 4
```

Four out five genes from the average linkage cluster are in bigger complete linkage. If I had more time I would reorder the data on disease state and then cell state to see if these are correlated with some of clusters.

# CML\_lab6

2226768Y

16 March 2016

```
setwd("C:/Users/Fran/Documents/Bioinformatics/SystemsBio/CML")
data.raw <- ReadAffy()
eset <- rma(data.raw)
```

Read in the CEL files in the directory, then normalize the data

```
## Background correcting
## Normalizing
## Calculating Expression

pheno <- read.csv("phenodata.csv", header= TRUE, stringsAsFactors = F )
# change sample names to same as data
sampleN <- paste0(t(as.data.frame(strsplit(pheno$Source.Name, " ")))[,1], ".CEL")
pheno$Source.Name <- sampleN

#Extract columns needed

cellPhase <- factor(pheno$Characteristics..cell.cycle.phase., levels = c("G0", "G1/S/G2/M"), labels=c("G0", "G1/S/G2/M"))
disease <- factor(pheno$Characteristics..disease.state., levels = c("chronic myelogenous leukemia (CML)", "Normal"))
CMLphase <- factor(pheno$Characteristics..cml.phase., levels = c("chronic", "accelerated"), labels=c("CP", "AP"))
phenoSubset<- data.frame(disease,cellPhase,CMLphase, row.names = sampleN)

# reorder the phenoSubset the same as eset
phenoSubset <- phenoSubset[rownames(pData(eset)),]
# Correct the phenotype data for CML subtype
correctPhase <- c(rep("CP",5),rep("AP",3),rep("",4),rep("CP",5) ,rep("AP",3),rep ("",4))
phenoSubset$CMLphase<- correctPhase

pheno.metadata <- data.frame(labelDescription=colnames(phenoSubset) )
pheno.annotatedDF <- new("AnnotatedDataFrame",data=phenoSubset, varMetadata=pheno.metadata)
phenoData(eset) <- pheno.annotatedDF
```

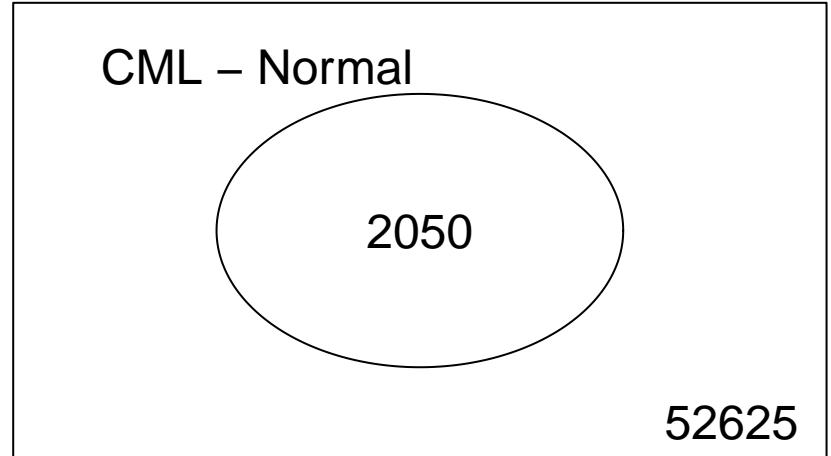
```
# create design array for all data where CML subset ==CP

goSamples <- pData(eset)$CMLphase != "AP" & pData(eset)$cellPhase == "G0"
eset <- eset[,goSamples]

design <- model.matrix(~0+ eset@phenoData$disease )
colnames(design) <- c("CML", "Normal")

contrast.matrix <- makeContrasts(CML-Normal,levels=design)
```

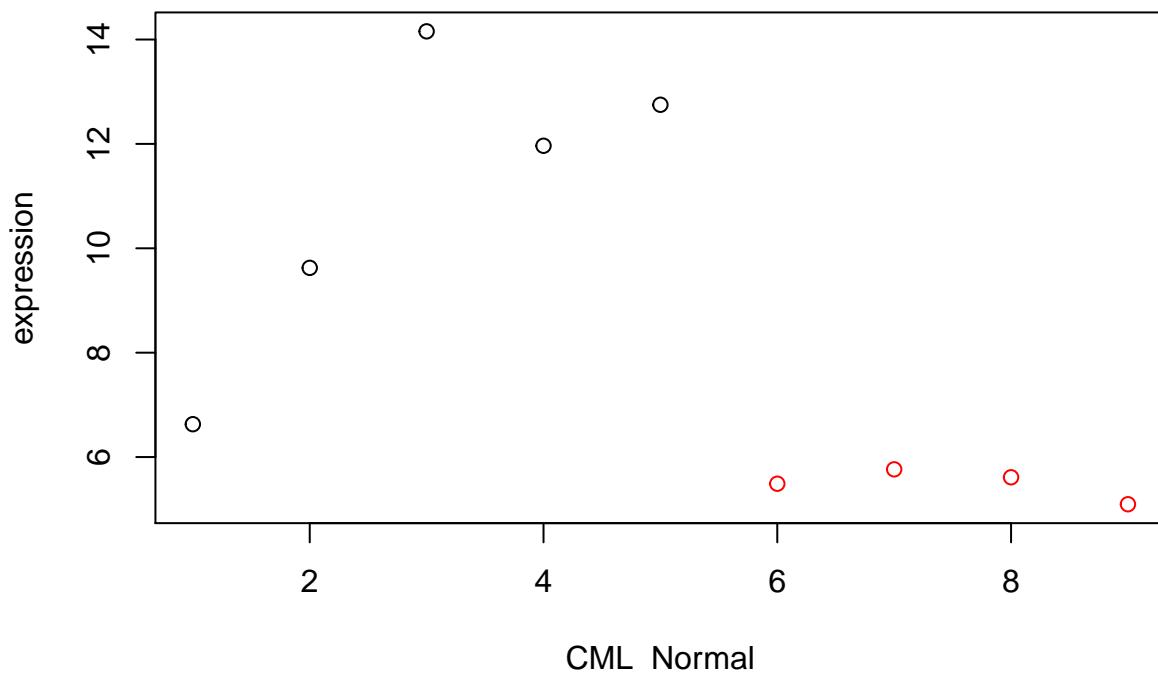
```
fit <- lmFit(eset,design)
fit2 <- contrasts.fit(fit, contrast.matrix)
fit2 <- eBayes(fit2)
results <- decideTests(fit2,adjust.method = "BH")
vennDiagram(results)
```



### Analysing the data

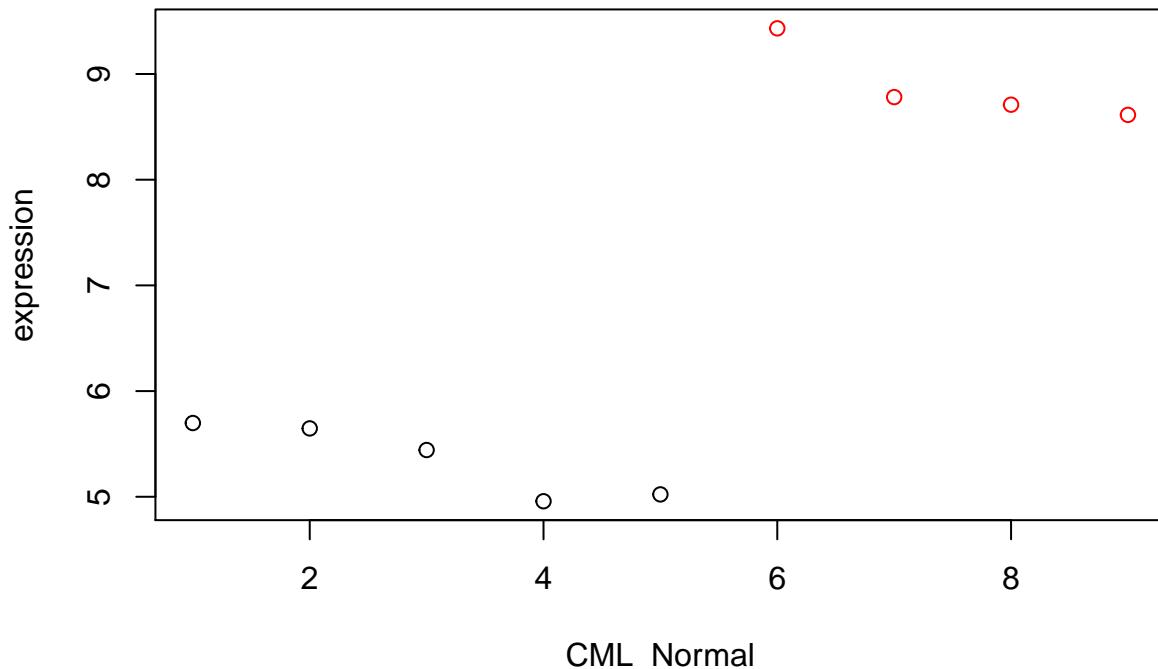
```
topGene <- rownames(topTable(fit2,sort.by = "logFC", number=1))
plot( exprs(eset)[topGene,], col=as.factor(pData(eset)$disease),main = " Most Differentially expressed genes")
```

## Most Differentially expressed gene by Fold change



```
topGeneP <- rownames(topTable(fit2, sort.by = "P", number=1))
plot( exprs(eset)[topGeneP,], col=as.factor(pData(eset)$disease), main = " Most Differentially expressed"
```

## Most Differentially expressed gene by adjusted p-value



## Part 2 ##The hypergeometric distribution 1. What is the dhyper() command you would execute in R to check this result? dhyper computes via binomial probabilities 2. What is the chance of selecting two aces if you are dealt 7 cards from a normal deck? What is the dhyper() command you would execute in R to calculate this?

```
dhyper (5,1000,30000-1000,50)
```

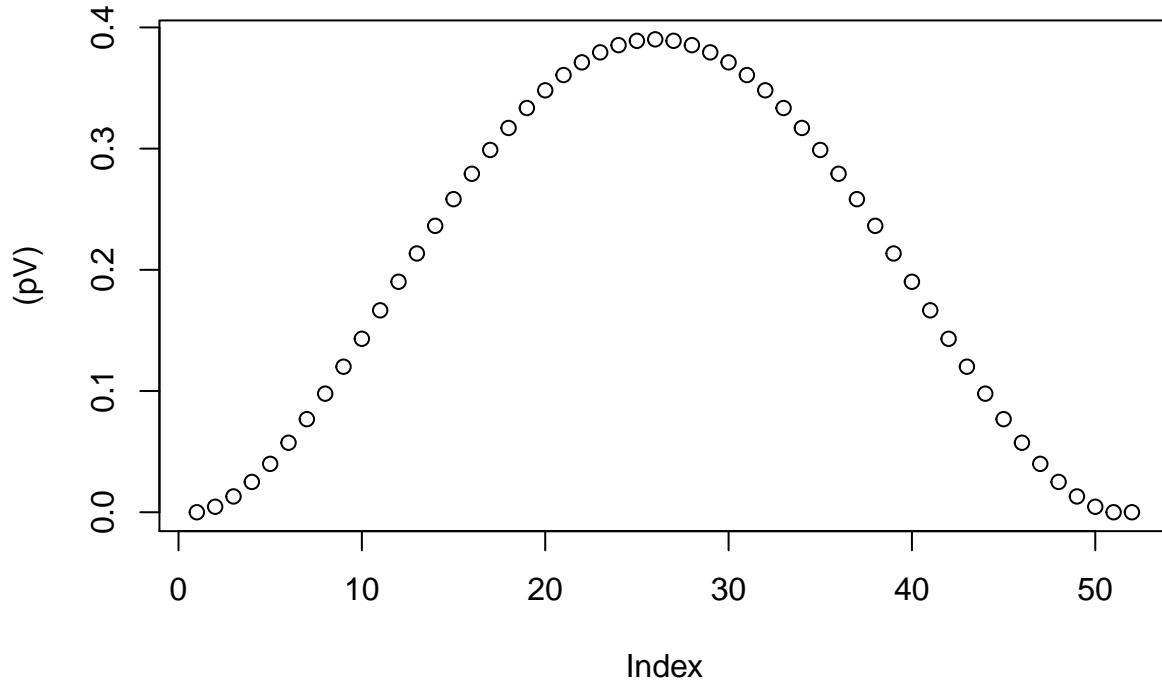
```
## [1] 0.0189012
```

```
#2/4 aces with 7 cards from 52 pack  
dhyper(2,4,48,7)
```

```
## [1] 0.07679379
```

3. What is the minimum number of cards you have to draw to have at least a 25% chance of having two aces? x = 2, 4 aces m=4, n= 52-4 =48, draws vector of possible draws.

```
draws <-c(1:52)  
pV <-dhyper (2,4, 48,draws)  
plot((pV))
```



```
which (pV > .25)
```

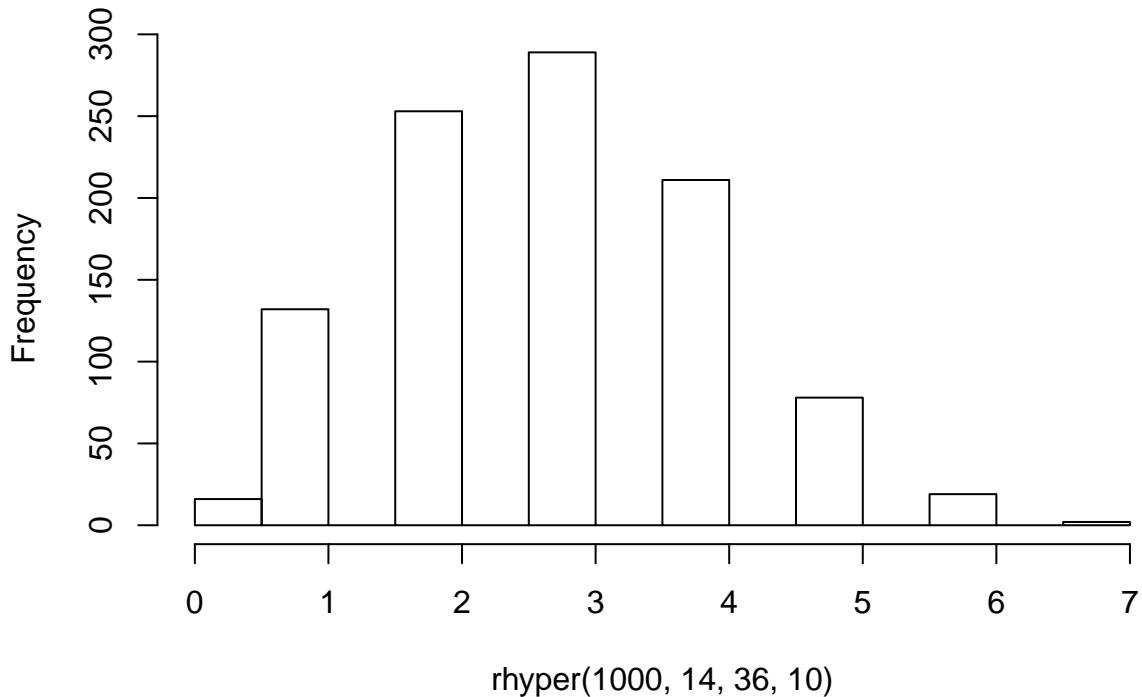
```
## [1] 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
```

15 is the minimum number of cards drawn to have a probability =.25 chance of drawing 2 aces.

4. Draw the histogram showing how many red suit cards (i.e., hearts or diamonds) with a value of 7 or greater you will draw in a hand of size 10 (show this for 1000 random draws). There are  $2 \times (13-6) = 14$  (m),  $52-14 = 36$  (n), drawn 10 times (k)

```
hist(rhyper (1000,14,36,10))
```

Histogram of rhyper(1000, 14, 36, 10)



```
####3}Gene ontology
```

```
goterms <- "GO:0004713"  
id <- "222258_s_at"
```

```
res <- select(hgfocus.db,id , "GO", "PROBEID")
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
res[1:4,]
```

```
##      PROBEID          GO EVIDENCE ONTOLOGY  
## 1 222258_s_at GO:0005092      IMP      MF  
## 2 222258_s_at GO:0005515      IPI      MF  
## 3 222258_s_at GO:0005634      IEA      CC  
## 4 222258_s_at GO:0005737      IDA      CC
```

```
dim(res)
```

```
## [1] 18  4
```

```

genesDE <- topTable(fit2,number= 2000, adjust.method ="fdr",p.value=0.05)
ids <- rownames(genesDE)
res <- select(hgfocus.db,ids , 'ENTREZID' , "PROBEID")

```

### Get Entrez IDs for the probes

```

## 'select()' returned 1:many mapping between keys and columns

# remove rows with na and use only unique IDs
row.has.na <- apply(res, 1, function(x){any(is.na(x))})
res <- (res[!row.has.na,])
selectedGenes <- unique(res$ENTREZID)

universeID <- select(hgfocus.db,featureNames(eset) , 'ENTREZID' , "PROBEID")

```

```

## 'select()' returned 1:many mapping between keys and columns

row.has.na <- apply(universeID, 1, function(x){any(is.na(x))})
universeID <- (universeID[!row.has.na,])
universeID <- unique(universeID$ENTREZID)

```

### The hypergeometric distribution

```

param <- new("GOHyperGParams", geneIds=selectedGenes,
            universeGeneIds= universeID,
            annotation="org.Hs.eg.db",
            ontology="BP",
            pvalueCutoff=0.01,
            conditional=FALSE,
            testDirection="over")
hyp <- hyperGTest(param)
sumTable <- summary(hyp)

kable((sumTable[order(-sumTable$OddsRatio),,drop =FALSE])[1:20,])

```

	GOBPID	Pvalue	OddsRatio	ExpCount	Count	Size	Term
157	GO:0002572	0.0047621	Inf	0.1381268	2	2	pro-T cell differentiation
158	GO:0009439	0.0047621	Inf	0.1381268	2	2	cyanate metabolic process
159	GO:0009440	0.0047621	Inf	0.1381268	2	2	cyanate catabolic process
160	GO:0010999	0.0047621	Inf	0.1381268	2	2	regulation of eIF2 alpha phosphorylation by hem
161	GO:0034124	0.0047621	Inf	0.1381268	2	2	regulation of MyD88-dependent toll-like receptor
162	GO:0034773	0.0047621	Inf	0.1381268	2	2	histone H4-K20 trimethylation
163	GO:0071733	0.0047621	Inf	0.1381268	2	2	transcriptional activation by promoter-enhancer
164	GO:0072301	0.0047621	Inf	0.1381268	2	2	regulation of metanephric glomerular mesangial
165	GO:0072602	0.0047621	Inf	0.1381268	2	2	interleukin-4 secretion
166	GO:0090202	0.0047621	Inf	0.1381268	2	2	gene looping

	GOBPID	Pvalue	OddsRatio	ExpCount	Count	Size	Term
167	<a href="#">GO:0090579</a>	0.0047621	Inf	0.1381268	2	2	dsDNA loop formation
168	<a href="#">GO:0097118</a>	0.0047621	Inf	0.1381268	2	2	neuroligin clustering involved in postsynaptic me
169	<a href="#">GO:1901301</a>	0.0047621	Inf	0.1381268	2	2	regulation of cargo loading into COPII-coated ve
170	<a href="#">GO:1902524</a>	0.0047621	Inf	0.1381268	2	2	positive regulation of protein K48-linked ubiquit
71	<a href="#">GO:0097368</a>	0.0012438	40.64200	0.2762535	3	4	establishment of Sertoli cell barrier
72	<a href="#">GO:2000630</a>	0.0012438	40.64200	0.2762535	3	4	positive regulation of miRNA metabolic process
18	<a href="#">GO:0046501</a>	0.0000074	27.22837	0.6215705	6	9	protoporphyrinogen IX metabolic process
106	<a href="#">GO:0006189</a>	0.0029497	20.31842	0.3453169	3	5	'de novo' IMP biosynthetic process
107	<a href="#">GO:0006526</a>	0.0029497	20.31842	0.3453169	3	5	arginine biosynthetic process
108	<a href="#">GO:0034201</a>	0.0029497	20.31842	0.3453169	3	5	response to oleic acid

Use hyperGTest, first set up parameters

# Assignment

2226768Y

19 March 2016

## Introduction

Using microarray data to find link between gene expression and phenotype is . Finding a key set of genes where changes in expression indicate the presence or absence of a condition. such as disease state or drug response.

Chronic Myeloid Leukemia is a disease that is successfully treated by the drug . Unfortunately some patients do not respond and others become resistant to the drug after a long period of successful treatment. Being able to predict the non-responders early gives the clinician the opportunity to try different options early . In this assignment I attempt to use two supervised learning classifiers, logistic regression and random forest, to produce models that can predict drug response from micro array data.

## The Data.

The dataset GSE14671 was downloaded from GEO at the NCBI website. It consists of 59 samples of microarray data each of 54675 probes, the metadata includes information on drug response. The sets were collected and processed at different times and the difference can be seen in the raw data.

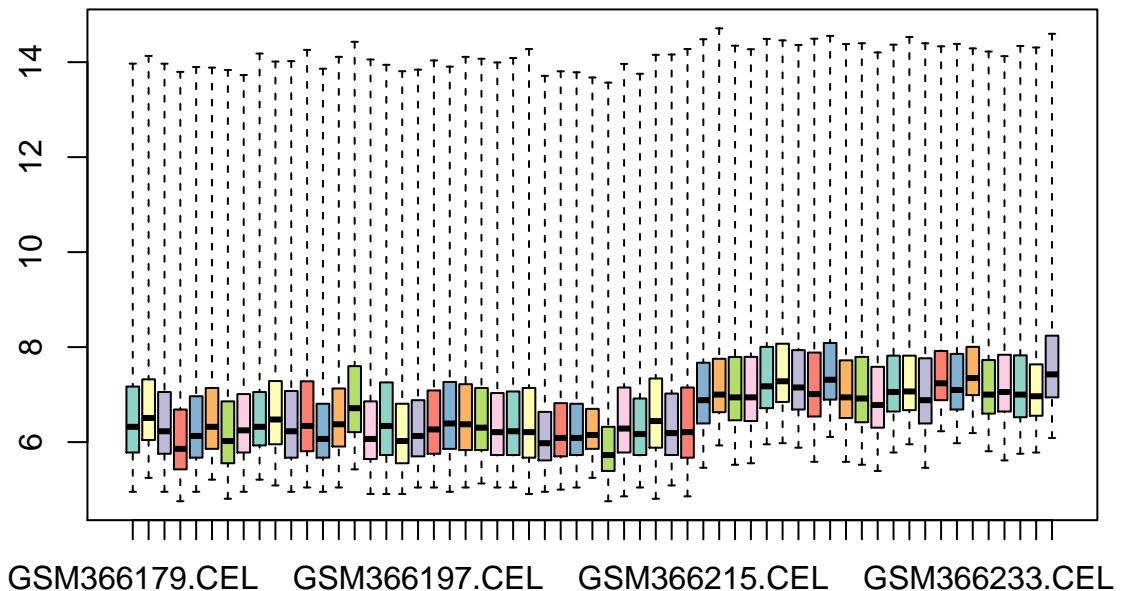
```
boxplot(data, main="Raw Data", col=brewer.pal(8,"Set3"))

## Warning: replacing previous import by 'utils::tail' when loading
## 'hgu133plus2cdf'

## Warning: replacing previous import by 'utils::head' when loading
## 'hgu133plus2cdf'

##
```

## Raw Data



```
#dat is an eset
ggboxplot <-function ( dat, title){
  library(reshape2)
  plotData <- as.data.frame(t(exprs(dat)))
  plotData$sample <- row.names(plotData)
  plotData.m <- melt(plotData, id.vars = "sample")

  g<- ggplot(plotData.m, aes(x=sample,y=value, fill= sample )) +
  geom_boxplot() +guides(fill=FALSE)+ ggttitle(title)

  return(g)
}
```

Function to plot boxplots of expression sets using ggplot

### Normalisation of the data

The data was normalised using the RMA routine in Affy. Robust Multi-Array Average, is a 3 pre-processing method. Background correction is applied to remove local artefacts and “noise”. Then the data is normalised across the samples to remove array effects, this means measurements from different arrays are comparable. Finally the data is summarised to combine probe intensities across arrays to give final expression values . (log2 values). The results can be seen in the boxplot below .

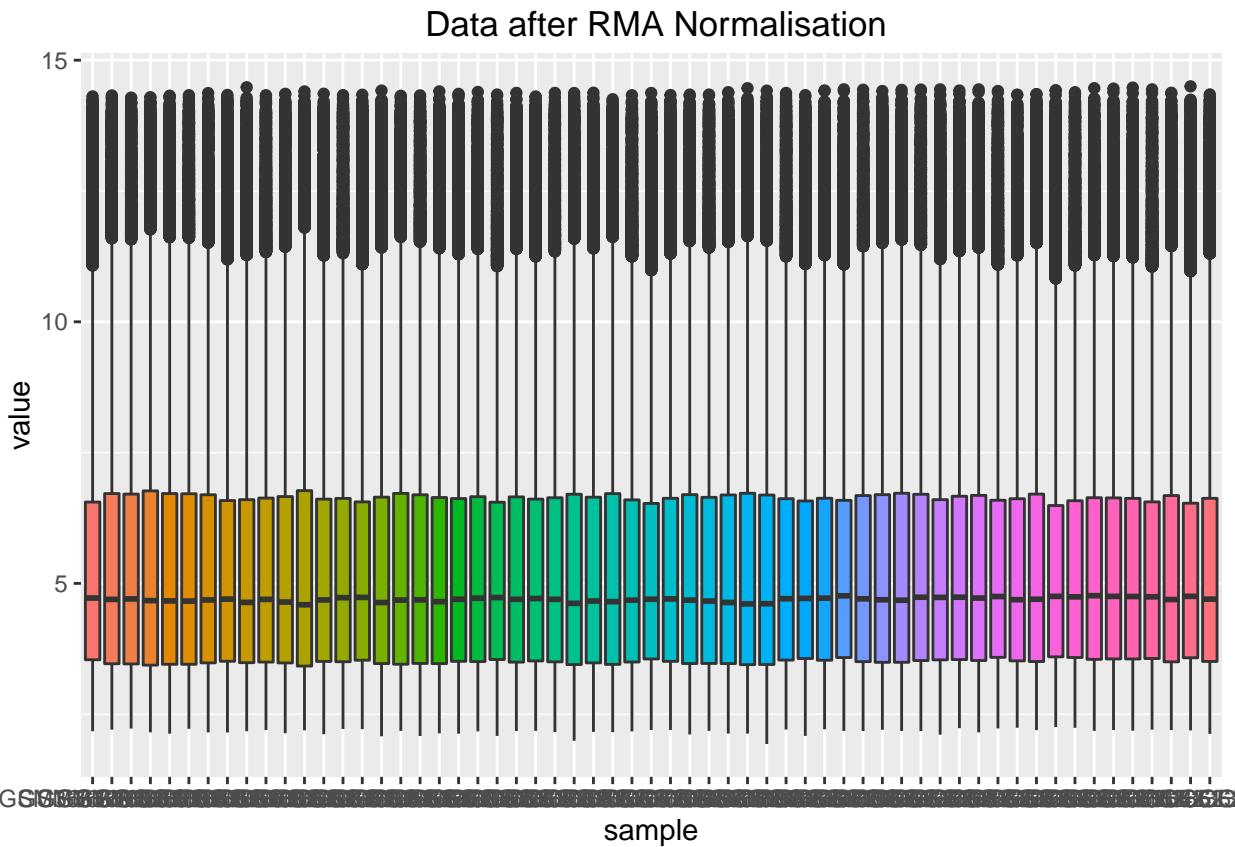
```

eset <- rma(data)

## Background correcting
## Normalizing
## Calculating Expression

ggboxplot (eset, " Data after RMA Normalisation")

```



Read in the phenotype data and extract columns needed and merge with the expression set.

```

phenoTypes <- read.table("E-GEO-14671.sdrf.txt", header = TRUE, sep= "\t", stringsAsFactors = F, row.names = 1)

pheno <- phenoTypes
# change sample names to same as data
sampleN <- paste0(t(as.data.frame(strsplit(rownames(pheno), " ")))[,1], ".CEL")

```

Extract columns needed response and training set

```

response <- factor(pheno$Comment.Sample_source_name., labels= c("NR","R"))
responseAll <- response # to try to get around knitting problem later on
tset<- data.frame(strsplit(pheno$Comment.Sample_description., " "),stringsAsFactors = FALSE)
tset <- paste(tset[1,],tset[2,],tset[3,] ,sep ="_")
trainingSet <- factor(tset, labels = c( "TS_R","TS_NR" , "VS_NR","VS_R"))
pheno<- data.frame(response,trainingSet, row.names = sampleN)

```

merge with pheno data from eset so in correct order

```
pd<- pData(eset)
pheno <- merge(pd, pheno, by=0)
rownames(pheno) <- pheno[, 'Row.names']

pheno.metadata <- data.frame(labelDescription=colnames(pheno) )
pheno.annotatedDF <- new("AnnotatedDataFrame", data=pheno, varMetadata=pheno.metadata)
phenoData(eset) <- pheno.annotatedDF
```

Re-order the eset so samples grouped by response.

```
cindex <- rownames( pData(eset) [order(pData(eset)$response), , drop = FALSE])
eset <- eset[, cindex]
```

## Data Exploration

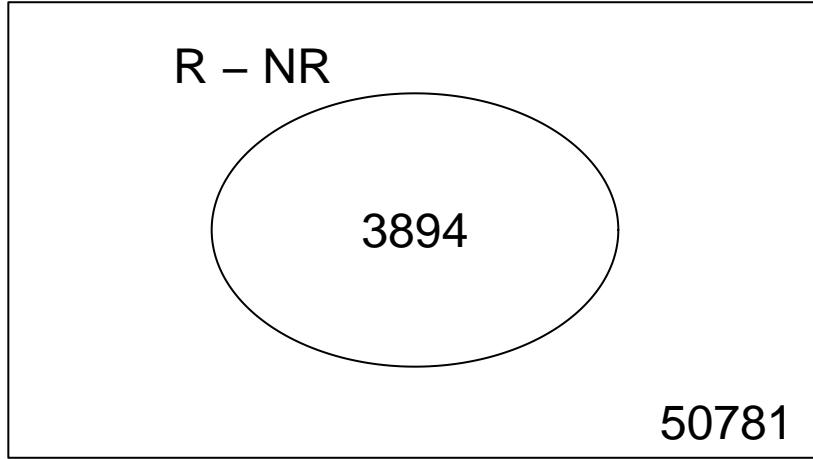
Limma was used to calculate the differential expression.

create design array for all data where CML subset ==CP.

```
design <- model.matrix(~0 + eset@phenoData$response )
colnames(design) <- c("R", "NR")
contrast.matrix <- makeContrasts(R-NR, levels=design)

fit <- lmFit(eset, design)
fit2 <- contrasts.fit(fit, contrast.matrix)
fit2 <- eBayes(fit2)

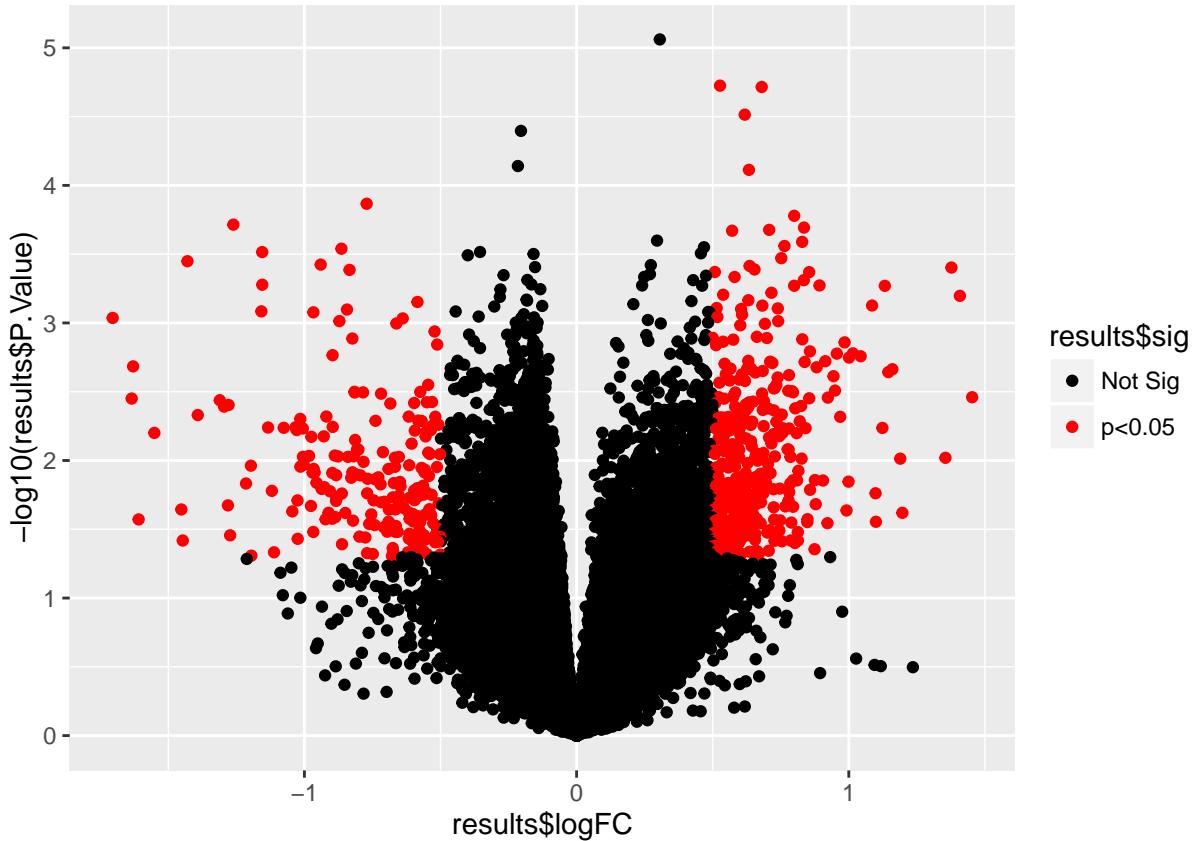
res = decideTests(fit2, adjust.method = "none", p.value= .05 )
vennDiagram(res, sub= "DE genes, with no multi-test correction")
```



### Volcano plot of results

The red genes have a p-value of less than 0.05 and a log2(fold-change) of greater than 0.5

```
results = topTable(fit2, number = Inf)
results = mutate(results, sig=ifelse((results$P.Value<0.05 & abs(results$logFC) > 0.5), "p<0.05", "Not Significant"))
p = ggplot(results, aes(results$logFC, -log10(results$P.Value))) +
  geom_point(aes(col=results$sig)) +
  scale_color_manual(values=c( "black","red"))
p
```



The p-values are high so if a multitest correction was applied no genes pass the test. Only 51 genes have a fold change of greater than 2 and p value of less than 0.05. ‘sum(results\$sig == “p<0.05”’ genes have a log2fold change = 0.5, (actual fold change of 1.4 and p<0.05).

## Feature Selection

To reduce the dimesionality curse of high feature number to low sample number differant ways were used in attempt to pick genes that might be relavent to the response. This also reduces the computational complexity of the problem for use on a PC.

- 1.Limma was used to produce a list of genes that showed a statistacally significant fold change .
- 2.Average Differance between responers and non- responders were calculated and then ranked on absolute value. It is assumed that DE is an affect of drug treatment.
- 3.Correlation between the change in gene expression and respone was calcvulated and ranked on absolute value. It is assumed that the expresion of importtant genes will correlate with response.

## Differentially expressed genes from Limma

```
tt <- topTable( fit2, adjust="none", n=Inf, sort.by="none" )
all.results.sorted <- tt[ order(abs(tt$P.Value)), ]

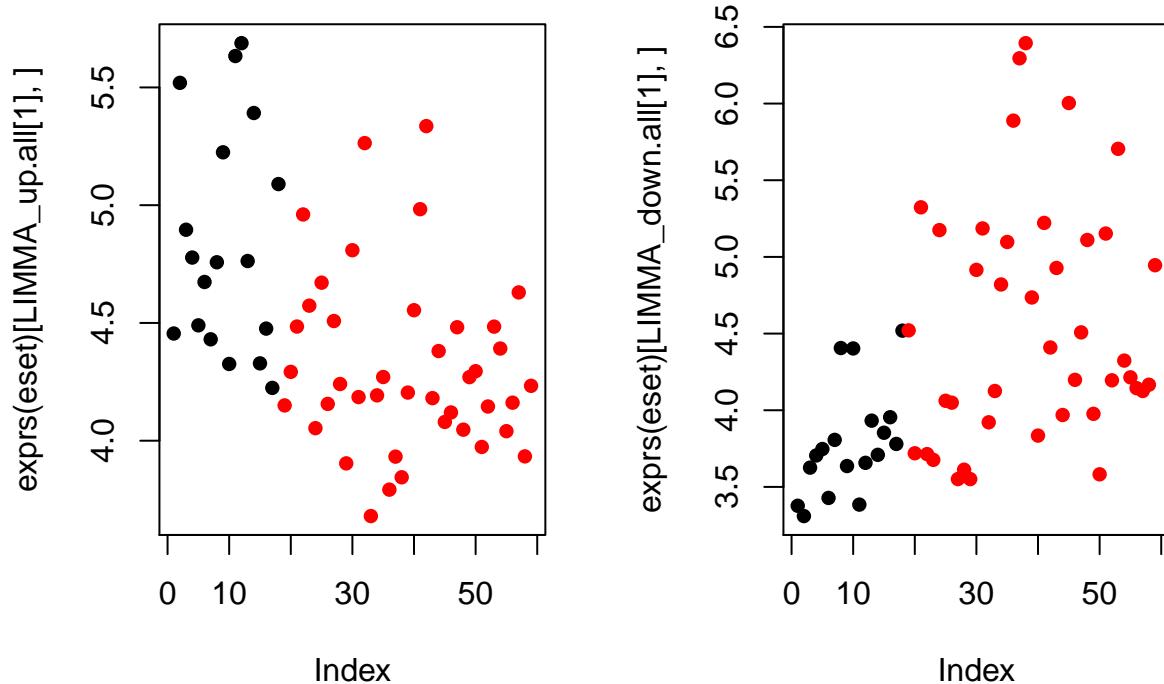
q.threshold <- 0.05
fc.threshold <- 0.5
LIMMA_up.all <- rownames( subset( all.results.sorted,
```

```

adj.P.Val<q.threshold & logFC>fc.threshold ) )
LIMMA_down.all <- rownames( subset( all.results.sorted,
adj.P.Val<q.threshold & logFC< (0-fc.threshold ) ))
Limma.all <- c(LIMMA_down.all,LIMMA_up.all)

par( mfrow=c(1,2) )
plot( exprs(eset)[LIMMA_up.all[1],],pch=16, col=as.factor(pData(eset)$response))
plot( exprs(eset)[LIMMA_down.all[1],],pch=16, col=as.factor(pData(eset)$response))

```

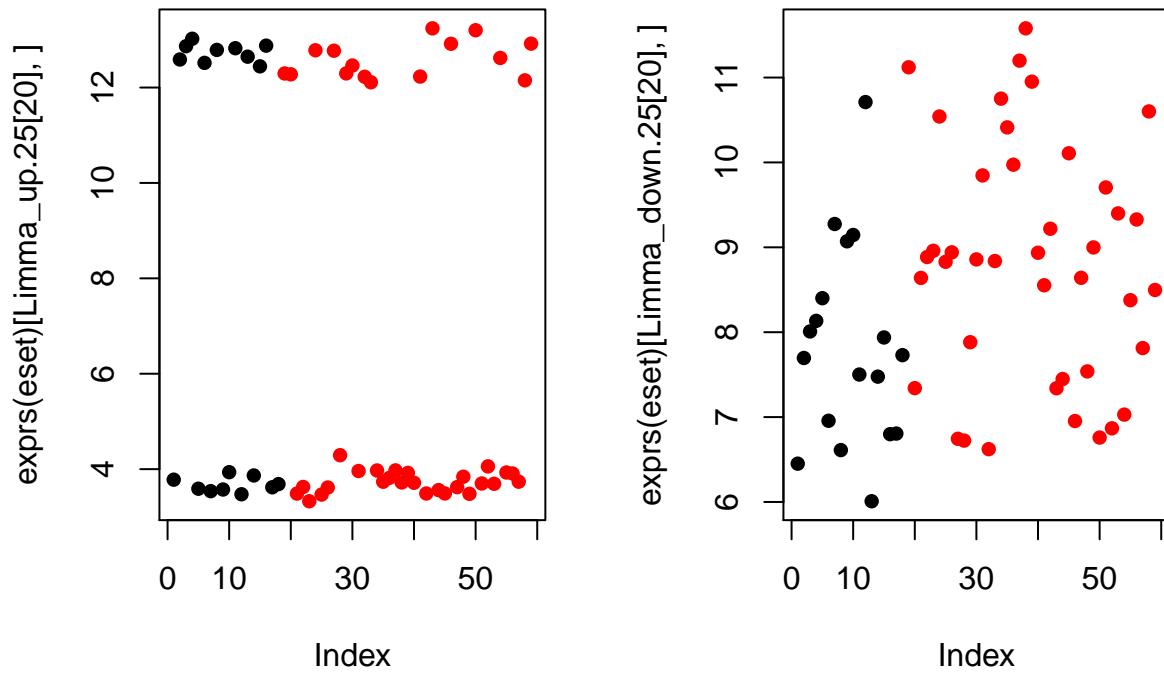


```

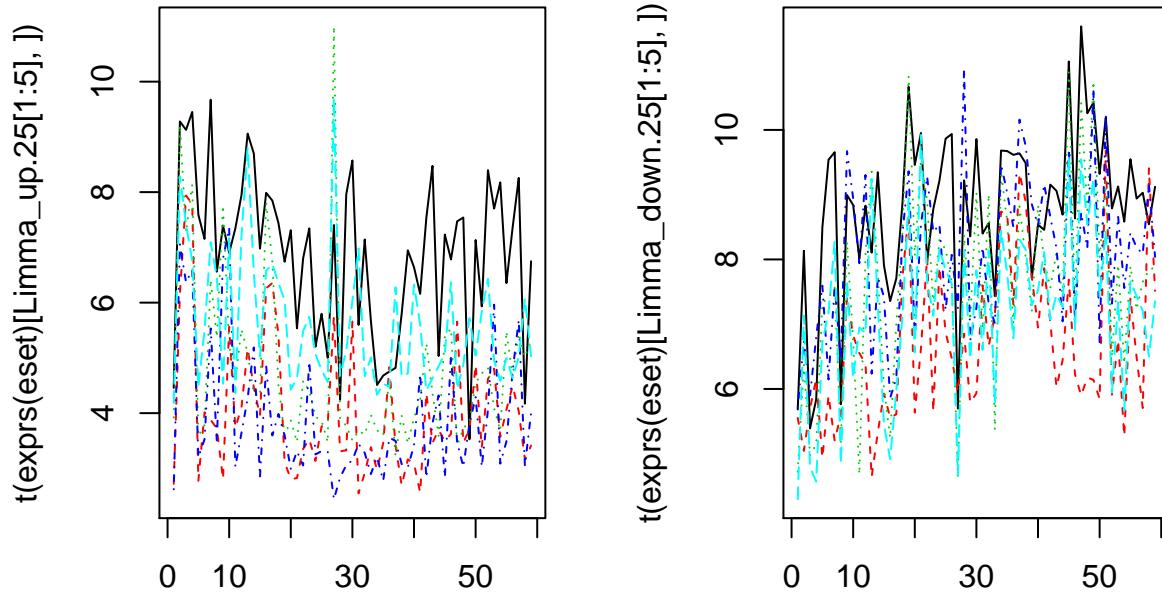
n.genes <- 25
Limma_up.25 <- rownames (subset(all.results.sorted, logFC>1 ))
Limma_down.25 <- rownames(subset(all.results.sorted, logFC< -1 )[1:n.genes,])
Limma50<- c(Limma_up.25,Limma_down.25)

plot( exprs(eset)[Limma_up.25[20],],pch=16, col=as.factor(pData(eset)$response))
plot( exprs(eset)[Limma_down.25[20],],pch=16, col=as.factor(pData(eset)$response))

```



```
matplot (t(exprs(eset)[Limma_up.25[1:5],]),type = "l")
matplot (t(exprs(eset)[Limma_down.25[1:5],]),type = "l")
```

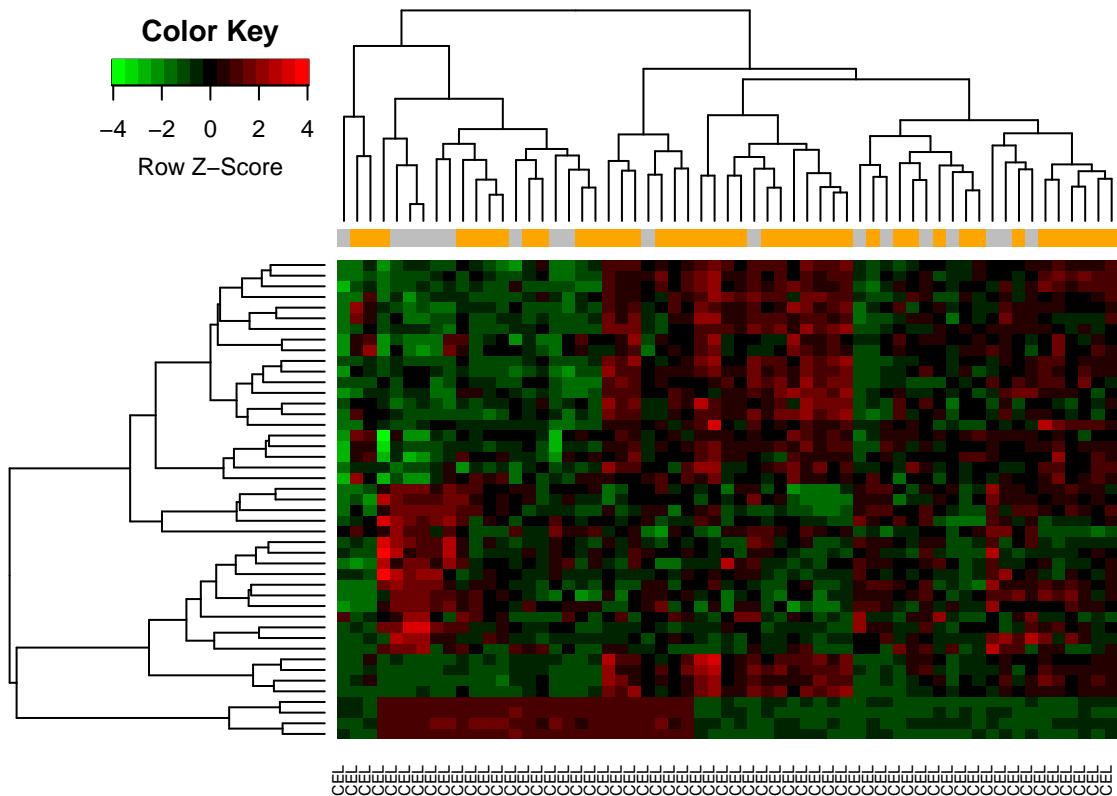


```
####Summary of the mean fold change.
All up regulated genes is: 0.6620217
Most significant up reg genes: 1.1679126 All down regulated genes: -0.7836567
Most significant down-reg genes:-1.2319727
```

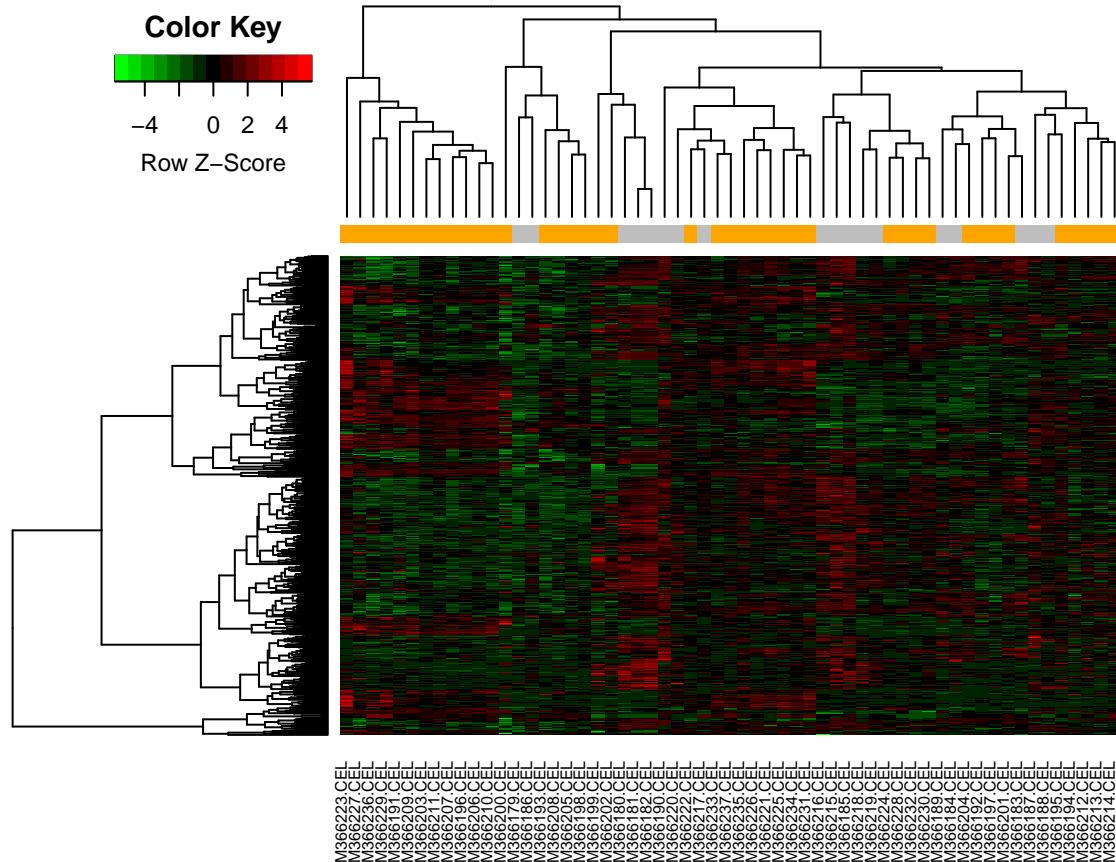
#### Heat map of most significant genes.

```
labels = paste0(pData(eset)$sample, " ",pData(eset)$response)
column.classification <- rep.int( "grey", ncol(eset))
column.classification[ pData(eset)$response=="R" ]<- "orange"

heatmap.2(exprs(eset) [Limma50,], scale="row",trace="none",density.info="none",col=greenred,labRow=NA,Col
```



```
heatmap.2(exprs(eset)[Limma.all,], scale="row",trace="none",density.info="none",
col=greenred,labRow=NA,ColSideColors=column.classification,key=T)
```



### ### Mean Difference in Expression

Using the difference between mean expression values for the responders and non-responders. This takes no account in the variance so that the DE of many genes may not be statistically significant.

```

expData <- exprs(eset)
responder <- (eset@phenoData$response == "R")
nonResponder <- (eset@phenoData$response != "R")
# avN <- mean(expData[1,normal])
#diff <- abs (avD-avN)
diff <- vector(, length = length(expData[,1]))
varG <- vector(, nrow(expData))
for (i in 1: length(expData[,1])){
  avNR<- mean(expData[i,nonResponder])
  avR <- mean(expData[i,responder])
  varG[i] <- var(expData[i,])
  diff[i] <- abs (avNR-avR)
}

rnames <- rownames(expData)
diffData <- data.frame (diff, row.names = rnames)

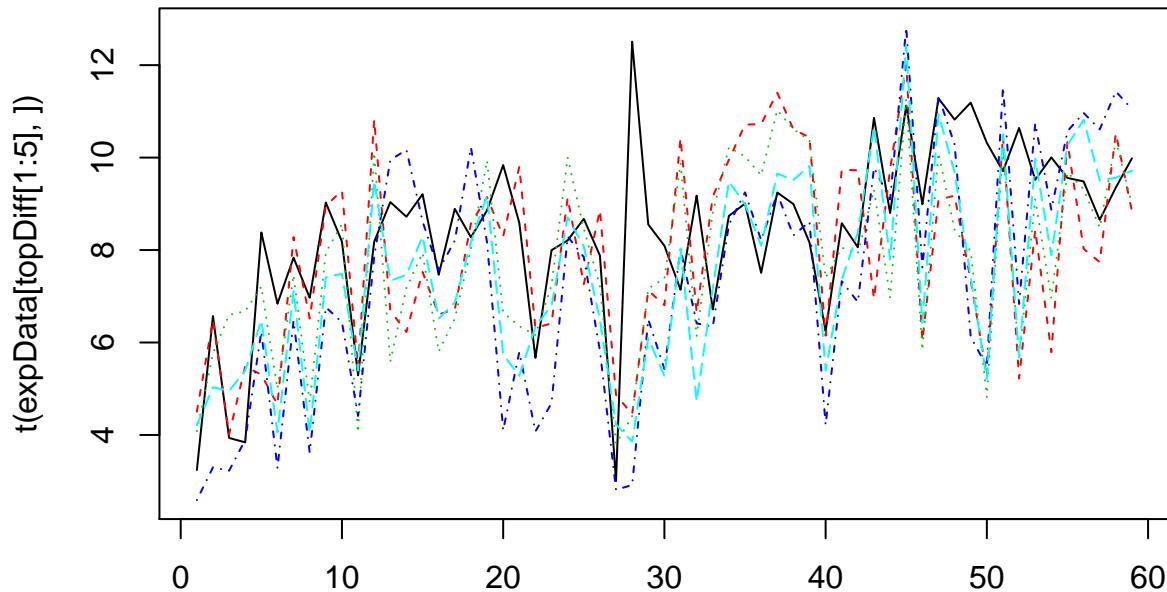
topDiff <- head (diffData[order(-diffData$diff), , drop = FALSE], n=50)
topDiff1000 <- head (diffData[order(-diffData$diff), , drop = FALSE], n=1000)
topDiff <- rownames (topDiff)
topDiff1000 <- rownames (topDiff1000)

```

Plot of expression against sample showing how even for the 5 genes with highest difference in mean the change

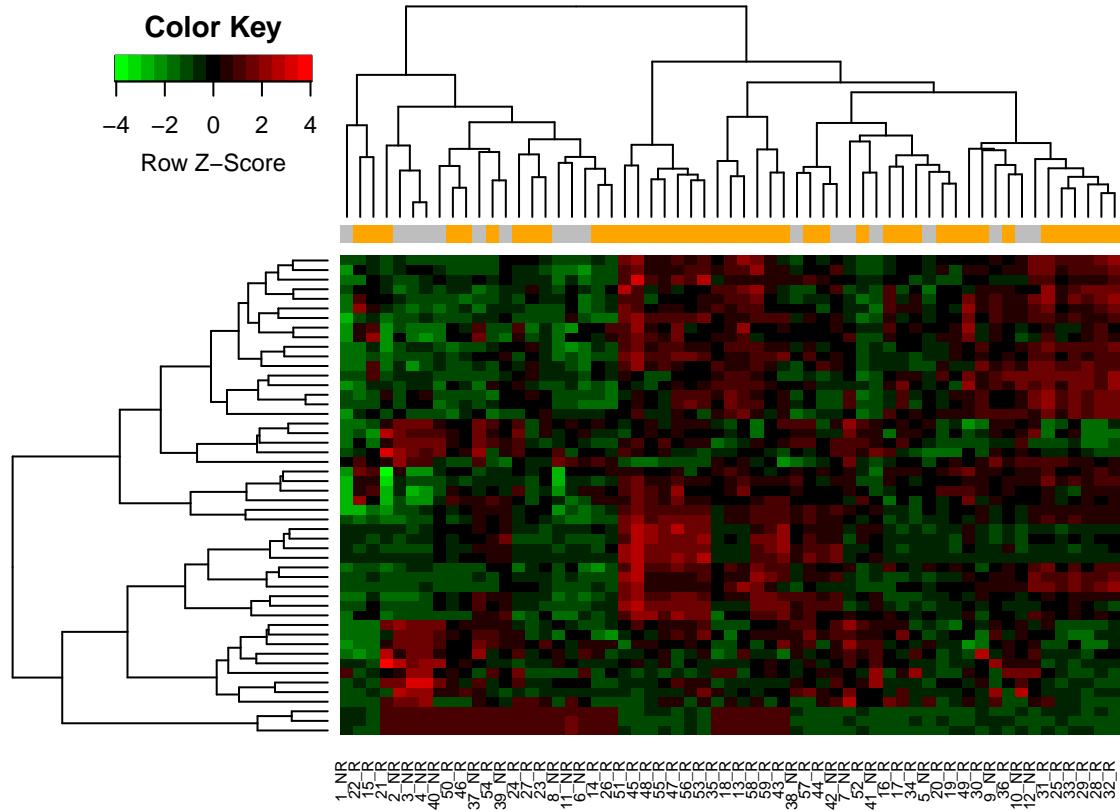
is very small compared to the variance.

```
matplot(t(expData [topDiff[1:5],]), type = "l")
```



##### Heatmap of the 1000 genes showing the greatest difference in mean expression.

```
heatmap.2(expData[topDiff,],scale="row",trace="none",density.info="none",col=greenred,  
labCol = labels,labRow=NA,ColSideColors=column.classification,key=T)
```



## Correlation

Choosing the genes that are most correlated to the response (either positive or negative).

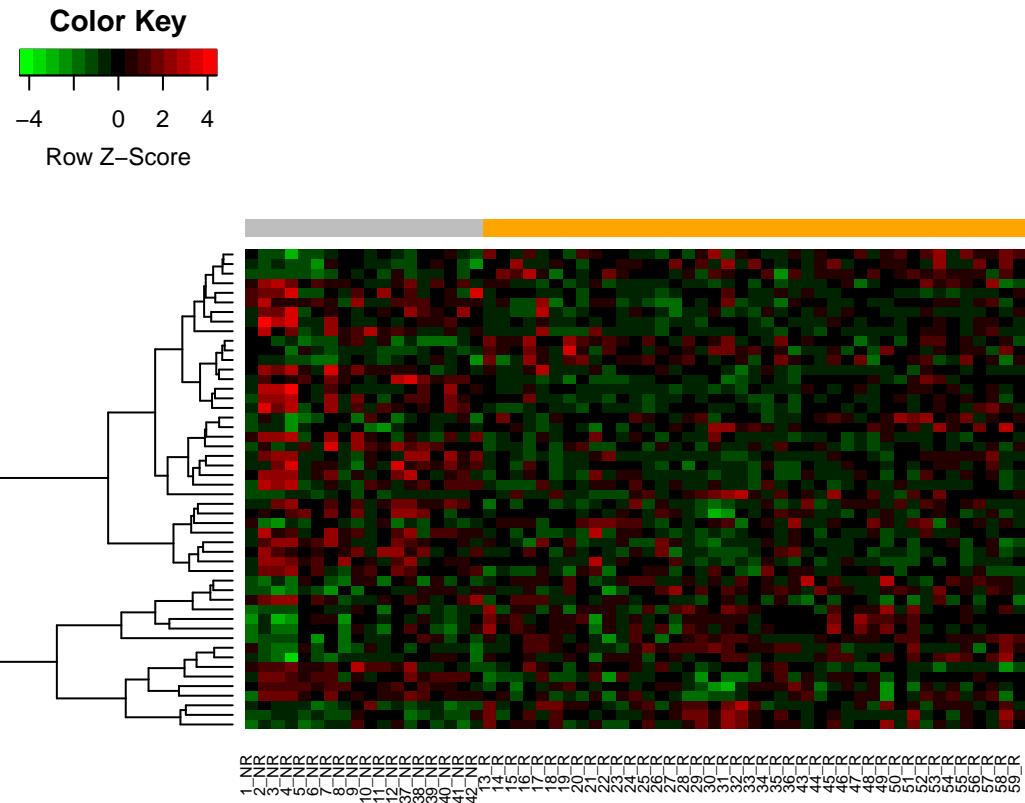
```

resp <- as.numeric(pData(eset)$response)
expData <- exprs(eset)

c<- cor (t(expData),resp)
#m[order(m[,1]),]
cc<- c[order(-abs(c[,1])), ]
c50 <- cc[1:50]
c1000 <- cc[1:1000]
corrGenes <- names(c50)
corr1000Genes <- names(c1000)
heatmap.2(expData[corrGenes,],
          scale="row",trace="none",density.info="none",col=greenred,Colv = FALSE,
          labCol = labels,labRow=NA,ColSideColors=column.classification,key=T)

## Warning in heatmap.2(expData[corrGenes, ], scale = "row", trace = "none", :
## Discrepancy: Colv is FALSE, while dendrogram is `both'. Omitting column
## dendrogram.

```

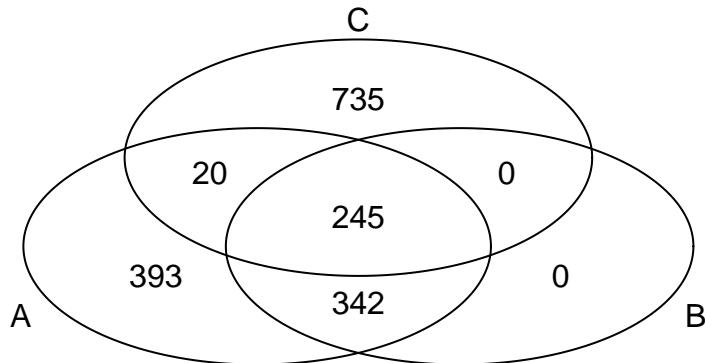


### Comparison between different genesets

Now have 3 different sets of genes generated in different ways, limma,corr and topDiff. These have been used to produce different size genesets , 1000 for random forest ( except limma that only had 587 genes statistically different with a fold change of over 1.4 ( $\log_2(0.5)$ ).with a subset of the top 50 to use with logistic regression . These were reduced more to see the effects on the results.

**Venn Diagram showing the separation of genesets** A= top average difference, B= Difference from Limma, C= Genes most correlated to response.

```
venn(list(topDiff1000,Limma.all,corr1000Genes))
```



## Predictive Modeling

### Aims

I aimed to see how much accuracy of prediction can be gained by using a more complex algorithm.  
I will use the following steps.

###Geneset Selection 1. By statistical significance. 2. By difference in expression. 3. By correlation.  
###Splitting data, by simple sampling, and with holding this data from training the model.

### ##Modeling

Logistic regression is used to determine how much variance in a binary dependent variable is explained by a set of independent variables. Logistic Regression does not make the same assumptions as linear regression such as linearity and homoscedasticity but does have a few key assumption that may not hold for gene expression data is that the data to be independent. The logit model was used that assumes that there is a log distribution of the probability of the event , “non responder”. The Random Forrest Classifier is an ensemble method that generate a multitude of decision trees in order to predict classes It has very few assumptions but is hard to interpret and therefore refine.

### ###Testing

Run the model on the withheld test data to estimate its accuracy.

### ###Evaluation.

Calculate the accuracy, False discovery rate , precision and recall from the prediction results.

## Training and Test Data

To estimate the predictive ability of the models produced, we need to test the model on data not used in training the model. Splitting the data into training and test samples is the easiest and most simplistic way to achieve this. This may well affect the potential of the model by reducing the amount of training data which is already small compared to the number of features being searched . A more sophisticated approach is leave-one-out cross-validation which is used by the Random Forest algorithm.

The data was split into training set and a test set in a proportion of 7:3 resulting in 41 (R:NR 28:13) samples in the training set and 18 (R:NR 13:5) in the test set.

```
d <- t(exprs(eset))
set.seed(1)
trainSet<-sample_frac(as.data.frame(d), 0.7)
sid<-(rownames(trainSet)) # because rownames() returns character
testSet<- as.data.frame(d[!(rownames(d) %in% sid),])
trainResponse <- pData(eset)[rownames(trainSet),]$response
testResponse <- pData(eset)[rownames(testSet),]$response
colnames(testSet) <- paste0("X",colnames(testSet))
allData <- data.frame(t(expData))
```

## Modeling

### Logistic Regression Modeling

#### 1. For correlated genes

```
genelist <- corrGenes
response <- trainResponse
d = data.frame(trainSet[,genelist])

vars = paste0('response ~ X', paste(genelist, collapse=' + X'))
LRcorr.model = glm(as.formula(vars), data=d, family=binomial(link='logit'))

reality <- pData(eset)$response
Reality_pred <- data.frame(reality, row.names = colnames(exprs(eset)))
corrPredA <- predict(LRcorr.model,allData , type= "response")

corrPredAll <- factor( (corrPredA > 0.5),labels = c("NR","R"))
RPtemp <- merge(Reality_pred,corrPredAll, by =0, all.x=TRUE)
Reality_pred$corrA <- RPtemp$y

corrPredT <- predict(LRcorr.model,testSet , type= "response")
corrPredTest <- factor( (corrPredT > 0.5),labels = c("NR","R"))
RPtemp <- merge(Reality_pred,corrPredTest, by =0, all.x=TRUE)
Reality_pred$corrT <- RPtemp$y
```

## 2. For Limma genes.

```
genelist <- Limma50
d = data.frame(trainSet[,genelist])

vars = paste0('response ~ X', paste(genelist, collapse=' + X'))
LRLimma.model = glm(as.formula(vars), data=d, family=binomial(link='logit'))

pred <- predict(LRLimma.model, allData, type= "response")
pred <- factor( (pred > 0.5),labels = c("NR","R"))
RPtemp <- merge(Reality_pred,pred, by =0, all.x=TRUE)
Reality_pred$LGLimmaA <- RPtemp$y

pred <- predict(LRLimma.model,testSet, type= "response")
pred <- factor( (pred > 0.5),labels = c("NR","R"))
RPtemp <- merge(Reality_pred,pred, by =0, all.x=TRUE)
Reality_pred$LGLimmaT <- RPtemp$y
```

## 3. For MeanDiff genes.

```
genelist <- topDiff
d = data.frame(trainSet[,genelist])

vars = paste0('response ~ X', paste(genelist, collapse=' + X'))
LRdiff.model = glm(as.formula(vars), data=d, family=binomial(link='logit'))

pred <- predict(LRdiff.model, allData, type= "response")
pred <- factor( (pred > 0.5),levels = c(FALSE,TRUE),labels = c("NR","R"))
RPtemp <- merge(Reality_pred,pred, by =0, all.x=TRUE)
Reality_pred$meandiffA <- RPtemp$y

pred <- predict(LRdiff.model, testSet, type= "response")
pred <- factor( (pred > 0.5),levels = c(FALSE,TRUE),labels = c("NR","R"))
RPtemp <- merge(Reality_pred,pred, by =0, all.x=TRUE)
Reality_pred$meandiffT <- RPtemp$y
```

## 4. First attempt to refine the model by using only the most important genes in new model.

```
genelist <- corrGenes
response <- trainResponse
d = data.frame(trainSet[,genelist])

anovaModel <- anova(LRcorr.model,test="Chisq")
head(anovaModel)

## Analysis of Deviance Table
##
```

```

## Model: binomial, link: logit
##
## Response: response
##
## Terms added sequentially (first to last)
##
##
##          Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## NULL              40    51.221
## X220979_s_at     1   10.4474    39    40.773 0.001228 **
## X40612_at        1    9.6075    38    31.166 0.001938 **
## X228411_at       1    4.6239    37    26.542 0.031529 *
## X1553572_a_at   1    6.5402    36    20.002 0.010546 *
## X239849_at       1    3.8441    35    16.158 0.049922 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# pick the significant genes
glmGenes <- row.names(anovaModel[anovaModel$`Pr(>Chi)` < .05, ])
glmGenes <- glmGenes [2:5]

newVars <- paste0('response ~ ', paste(glmGenes, collapse=' +'))
newModel = glm(as.formula(newVars), data=d, family=binomial(link='logit'))
summary(newModel)

##
## Call:
## glm(formula = as.formula(newVars), family = binomial(link = "logit"),
##      data = d)
##
## Deviance Residuals:
##      Min        1Q        Median         3Q        Max 
## -2.05087  -0.04957   0.04913   0.40788   1.75973 
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -47.13940  38.29697 -1.231  0.2184    
## X220979_s_at -8.78028   3.57597 -2.455  0.0141 *  
## X40612_at     -2.09600   1.36816 -1.532  0.1255    
## X228411_at     -0.01987   1.09878 -0.018  0.9856    
## X1553572_a_at 13.95733   6.88461  2.027  0.0426 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 51.221  on 40  degrees of freedom
## Residual deviance: 20.002  on 36  degrees of freedom
## AIC: 30.002
##
## Number of Fisher Scoring iterations: 7

```

```

anovaModel <-anova(newModel,test="Chisq")
head(anovaModel)

## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: response
##
## Terms added sequentially (first to last)
##
##
##          Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## NULL             40      51.221
## X220979_s_at    1   10.4474     39    40.773 0.001228 **
## X40612_at       1    9.6075     38    31.166 0.001938 **
## X228411_at       1    4.6239     37    26.542 0.031529 *
## X1553572_a_at   1    6.5402     36    20.002 0.010546 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

newPred <- predict(newModel, allData, type= "response")
newPred <-factor( (newPred > 0.5),labels = c("NR","R"))
Reality_pred$LGcorrImpA <- newPred

pred <- predict(newModel, testSet, type= "response")
pred <- factor( (pred > 0.5),levels = c(FALSE,TRUE),labels = c("NR","R"))
RPtemp <- merge(Reality_pred,pred, by =0, all.x=TRUE)
Reality_pred$LGcorrImpT <-RPtemp$y

```

Functions to calculate the accuracy,FDR,precision and recall from the confusion matrix.

```

accFDR <- function (TP,FP,TN,FN){
  Acc <- (TN+TP)/(TP+TN+FP+FN)
  Pre <- TP/(TP+FP)
  Recall <- TP/(TP+FN)
  FDR <- FP/(FP+TN)
  resList <- list(c(Acc,FDR,Pre,Recall,TP,FP,TN,FN))
  return (resList)
}

```

Function Calculate Accuracy,FDR, recall and precision from TP, FP, FN, TN for each model

```

confuse <- function(rp) {
  res <- list("")
  resultList <- list ("")
  predNames <- colnames(rp)
  for (x in 1:(ncol(rp)-1)){
    predName <- predNames[x+1]

```

```

realPred <- rp[, c("reality" ,predName)]
# final[complete.cases(final),]
colnames (realPred) <- c("reality","prediction")
realPred <- realPred[complete.cases(realPred),]
TP<-FP<-FN<-TN <-0
for (i in 1:nrow(realPred)){
  if (realPred$reality[i]== "NR" && realPred$prediction[i]== "NR")
    TP =TP +1
  else if (realPred$reality[i]== "NR" && realPred$prediction[i]== "R")
    FN = FN +1
  else if (realPred$reality[i]== "R" && realPred$prediction[i]== "NR")
    FP = FP +1
  else if (realPred$reality[i]== "R" && realPred$prediction[i]== "R")
    TN = TN +1
}
resAcc <- accFDR(TP,FP,TN,FN)
resultList <- append(resultList,resAcc)
}
return(resultList)
}

```

## Randomn Forest

### 1.Correlated geneset

```

response <- pData(eset)$response
subsetGenes <- corr1000Genes
trainSubset <- t(expData[subsetGenes,])

fitRF.corr <- randomForest(x= trainSubset, y=response ,importance=TRUE ,ntree=5001, proximity=TRUE)
confusion <- fitRF.corr$confusion
accRes<- accFDR (confusion[1,1],confusion[2,1],confusion[2,2],confusion[1,2])
RFresults <- data.frame(accRes[[1]])
colnames(RFresults) = ("RFcorrA")

```

top 1000 correlated geneset

```

subsetGenes <- corrGenes
trainSubset <- t(expData[subsetGenes,])
fitRF.corr <- randomForest(x= trainSubset, y=response ,importance=TRUE ,ntree=5001, proximity=TRUE)
confusion <- fitRF.corr$confusion
accRes<- accFDR (confusion[1,1],confusion[2,1],confusion[2,2],confusion[1,2])
RFresults$RFcorr50 <- accRes[[1]]

```

Top 50 correlated geneset

```

subsetGenes <- corrGenes[1:6]
trainSubset <- t(expData[subsetGenes,])
fitRF.corr <- randomForest(x= trainSubset, y=response ,importance=TRUE ,ntree=5001, proximity=TRUE)
confusion <- fitRF.corr$confusion
accRes<- accFDR (confusion[1,1],confusion[2,1],confusion[2,2],confusion[1,2])
RFresults$RFcorr6 <- accRes[[1]]

```

Top 6 correlated genes

## 2.Limma Geneset

```

subsetGenes <- Limma.all
trainSubset <- t(expData[subsetGenes,])

# Fitting model
fitRF.limma <- randomForest(x= trainSubset, y=response ,importance=TRUE ,ntree=5001, proximity=TRUE)

confusion <- fitRF.limma$confusion
accRes<- accFDR (confusion[1,1],confusion[2,1],confusion[2,2],confusion[1,2])
RFresults$RFlimmaAll <- accRes[[1]]

```

Limma all geneset (567 genes)

```

subsetGenes <- Limma50
trainSubset <- t(expData[subsetGenes,])

# Fitting model
fitRF.limma <- randomForest(x= trainSubset, y=response ,importance=TRUE ,ntree=5001, proximity=TRUE)

confusion <- fitRF.limma$confusion
accRes<- accFDR (confusion[1,1],confusion[2,1],confusion[2,2],confusion[1,2])
RFresults$RFlimma50 <- accRes[[1]]

```

Limma 50 geneset

```

subsetGenes <- Limma50[1:10]
trainSubset <- t(expData[subsetGenes,])

# Fitting model
fitRF.limma <- randomForest(x= trainSubset, y=response ,importance=TRUE ,ntree=5001, proximity=TRUE)

confusion <- fitRF.limma$confusion
accRes<- accFDR (confusion[1,1],confusion[2,1],confusion[2,2],confusion[1,2])
RFresults$RFlimma10 <- accRes[[1]]

```

## Limma 10 geneset

### 3.Mean Difference Geneset

```
subsetGenes <- topDiff1000
trainSubset <- t(expData[subsetGenes,])
fitRF.meanD <- randomForest(x= trainSubset, y=response ,importance=TRUE ,ntree=5001, proximity=TRUE)
confusion <- fitRF.meanD$confusion
accRes<- accFDR (confusion[1,1],confusion[2,1],confusion[2,2],confusion[1,2])
RFresults$RFtopD1000 <- accRes[[1]]
```

top 1000 geneset

```
subsetGenes <- topDiff1000[1:50]
trainSubset <- t(expData[subsetGenes,])
fitRF.meanD <- randomForest(x= trainSubset, y=response ,importance=TRUE ,ntree=5001, proximity=TRUE)
confusion <- fitRF.meanD$confusion
accRes<- accFDR (confusion[1,1],confusion[2,1],confusion[2,2],confusion[1,2])
RFresults$RFtopD50 <- accRes[[1]]
```

top 20 geneset

```
subsetGenes <- topDiff[1:10]
trainSubset <- t(expData[subsetGenes,])
fitRF.meanD <- randomForest(x= trainSubset, y=response ,importance=TRUE ,ntree=5001, proximity=TRUE)
confusion <- fitRF.meanD$confusion
accRes<- accFDR (confusion[1,1],confusion[2,1],confusion[2,2],confusion[1,2])
RFresults$RFtopD10 <- accRes[[1]]
```

Top 10 of mean difference geneset

## Results

```
LGResults <- confuse (Reality_pred)
resNames <- colnames(Reality_pred)
LGtable <- data.frame(LGResults[[2]])
for (i in 3:(length(LGResults)))
  LGtable <- cbind(LGtable,LGResults[[i]])
rownames(LGtable) <- c( "Accuracy","FDR", "Precision", "Recall","TP","FP","TN","FN" )
colnames(LGtable) <- resNames[2:length(resNames)]
LGtable <- t(LGtable)
```

## Logistic Regression

```
kable (LGtable, digits = 2)
```

	Accuracy	FDR	Precision	Recall	TP	FP	TN	FN
corrA	0.69	0.27	0.50	0.61	11	11	30	7
corrT	0.44	0.55	0.33	0.43	3	6	5	4
LGLimmaA	0.78	0.17	0.63	0.67	12	7	34	6
LGLimmaT	0.72	0.18	0.67	0.57	4	2	9	3
meandiffA	0.76	0.17	0.61	0.61	11	7	34	7
meandiffT	0.67	0.18	0.60	0.43	3	2	9	4
LGcorrImpA	0.90	0.05	0.88	0.78	14	2	39	4
LGcorrImpT	0.83	0.00	1.00	0.57	4	0	11	3

## Random Forest

```
rownames(RFresults) <- c( "Accuracy", "FDR", "Precision", "Recall", "TP", "FP", "TN", "FN" )
RFresults <- t(RFresults)
kable (RFresults, digits = 2)
```

	Accuracy	FDR	Precision	Recall	TP	FP	TN	FN
RFcorrA	0.80	0.02	0.88	0.39	7	1	40	11
RFcorr50	0.86	0.02	0.92	0.61	11	1	40	7
RFcorr6	0.88	0.05	0.87	0.72	13	2	39	5
RFlimmaAll	0.75	0.07	0.67	0.33	6	3	38	12
RFlimma50	0.80	0.07	0.75	0.50	9	3	38	9
RFlimma10	0.76	0.10	0.67	0.44	8	4	37	10
RFtopD1000	0.73	0.07	0.62	0.28	5	3	38	13
RFtopD50	0.76	0.12	0.64	0.50	9	5	36	9
RFtopD10	0.73	0.17	0.56	0.50	9	7	34	9

## Discussion

All the plots of expression by sample show how noisy the genes are compared to the fold change. This together with the lack of sample clustering in the heatmaps meant the expectations of building a reasonably accurate predictive model were quite low. The LR models were all built just using the training subset of data, but predictions were made on both the test data *.T* and on all the data A to try and get improve the accuracy of the accuracy/FDR data. (This is maybe completely invalid). The accuracy of all the models is very poor and the FDR is high so these would be useless as predictors of drug response where it is important to pick up all the non-responders. In this scenario it is important to have a very low FDR , to make sure all the non-responders are identified at the expense of falsely identifying responders.

The improvements made by picking only the 4 most important genes is very dramatic with accuracy rising (for just the test set) from 0.44 to 0.83 (0.93) and the FDR going from 0.55 to 0. (0.05 for all data). Considering the crude method that the initial geneset was selected and the low signal to noise ratio of the data this seems quite significant.

The results from the RF models show generally a higher accuracy using 50 rather than 1000 (587 for Limma)

but then when reducing this to 10 reducing the accuracy. The use of 6 genes for the correlation genest was chosen by trial and error to get the best improvement without increasing the FDR. Although the FDR has risen it may not be statistically significant as it went from 1 to 2 so may be part of the distribution of the data.

First we should check if the most important genes in each geneset overlap within each LG model and how many of the genes are in common. It may also be interesting to pick genesets by an unsupervised clustering method to compare the results.

Tuning the models, this was the first step in developing a model ad it can be seen by the improvement in LG for the correlation data the improvement. It would be interesting to see what further improvements were possible and what would happen if you combined the most important genes from each geneset.

Tuning the random forest model is more complicated because we do not get given a list of most important genes because different variables are important in different regions of the search. This means tuning is not a straight forward process but RF models are known to be greatly enhanced by removing the unimportant variables.

In conclusion it looks like both methods have the potential to produce an accurate predictor, further refinement of both models would be needed. Unfortunately I did not have time to refine the random forest models. More data should be used to verify any model produced.

## References

- [1] Shannon K. McWeney, Lucy C. Pemberton, Marc M. Loriaux, Kristina Vartanian, Stephanie G. Willis, Gregory Yochum, Beth Wilmot, Yaron Tur- paz, Raji Pillai, Brian J. Druker, Jennifer L. Snead, Mary MacPartlin, Stephen G. O'Brien, Junia V. Melo, Thoralf Lange, Christina A. Harring- ton, and Michael W. N. Deininger. A gene expression signature of cd34+ cells to predict major cytogenetic response in chronic-phase chronic myeloid leukemia patients treated with imatinib. *Blood*, 115(2):315{325, 2010.

## Predictions for LG modeling

```
kable(Reality_pred)
```

	reality	corrA	corrT	LGLimmaA	LGLimmaT	meandiffA	meandiffT	LGcorrImpA	LGcorr
GSM366179.CEL	NR	NR	NA	NR	NA	NR	NA	R	NA
GSM366180.CEL	NR	NR	NR	R	R	NR	NR	NR	NR
GSM366181.CEL	NR	NR	NA	NR	NA	NR	NA	NR	NA
GSM366182.CEL	NR	NR	NA	NR	NA	NR	NA	NR	NA
GSM366183.CEL	NR	NR	NA	NR	NA	NR	NA	NR	NA
GSM366184.CEL	NR	NR	NR	NR	NR	R	R	NR	NR
GSM366185.CEL	NR	R	R	NR	NR	R	R	NR	NR
GSM366186.CEL	NR	R	R	NR	NR	NR	NR	NR	NR
GSM366187.CEL	NR	NR	NA	NR	NA	NR	NA	NR	NA
GSM366188.CEL	NR	NR	NA	NR	NA	NR	NA	R	NA
GSM366189.CEL	NR	NR	NA	NR	NA	NR	NA	NR	NA
GSM366190.CEL	NR	NR	NA	NR	NA	NR	NA	NR	NA
GSM366215.CEL	NR	R	NA	R	NA	R	NA	NR	NA
GSM366216.CEL	NR	R	R	NR	NR	NR	NR	NR	R
GSM366217.CEL	NR	R	NA	R	NA	R	NA	R	NA
GSM366218.CEL	NR	R	NA	R	NA	R	NA	R	NA
GSM366219.CEL	NR	NR	NR	R	R	R	R	NR	R
GSM366220.CEL	NR	R	R	R	R	R	R	NR	R
GSM366191.CEL	R	R	NA	R	NA	R	NA	R	NA

	reality	corrA	corrT	LGLimmaA	LGLimmaT	meandiffA	meandiffT	LGcorrImpA	LGcorr
GSM366192.CEL	R	NR	NR	R	R	R	R	R	R
GSM366193.CEL	R	R	NA	R	NA	R	NA	R	NA
GSM366194.CEL	R	R	NA	R	NA	R	NA	R	NA
GSM366195.CEL	R	R	NA	R	NA	R	NA	R	NA
GSM366196.CEL	R	NR	NR	R	R	R	R	R	R
GSM366197.CEL	R	R	NA	R	NA	R	NA	R	NA
GSM366198.CEL	R	R	NA	R	NA	R	NA	R	NA
GSM366199.CEL	R	R	NA	R	NA	R	NA	NR	NA
GSM366200.CEL	R	R	NA	R	NA	R	NA	R	NA
GSM366201.CEL	R	R	NA	R	NA	R	NA	R	NA
GSM366202.CEL	R	R	NA	R	NA	R	NA	R	NA
GSM366203.CEL	R	R	NA	R	NA	R	NA	R	NA
GSM366204.CEL	R	R	NA	R	NA	R	NA	R	NA
GSM366205.CEL	R	R	NA	R	NA	R	NA	R	NA
GSM366206.CEL	R	R	R	R	R	NR	NR	R	R
GSM366207.CEL	R	R	NA	R	NA	R	NA	R	NA
GSM366208.CEL	R	R	NA	R	NA	R	NA	R	NA
GSM366209.CEL	R	NR	NA	NR	NA	NR	NA	R	NA
GSM366210.CEL	R	NR	NA	NR	NA	NR	NA	R	NA
GSM366211.CEL	R	R	R	R	R	R	R	R	R
GSM366212.CEL	R	NR	NA	NR	NA	NR	NA	R	NA
GSM366213.CEL	R	NR	NA	NR	NA	NR	NA	R	NA
GSM366214.CEL	R	NR	NA	NR	NA	NR	NA	R	NA
GSM366221.CEL	R	R	R	NR	NR	NR	NR	R	R
GSM366222.CEL	R	R	R	R	R	R	R	R	R
GSM366223.CEL	R	R	R	R	R	R	R	R	R
GSM366224.CEL	R	NR	NR	NR	NR	R	R	R	R
GSM366225.CEL	R	NR	NR	R	R	R	R	R	R
GSM366226.CEL	R	R	NA	R	NA	R	NA	R	NA
GSM366227.CEL	R	R	NA	R	NA	R	NA	R	NA
GSM366228.CEL	R	NR	NR	R	R	R	R	R	R
GSM366229.CEL	R	R	NA	R	NA	R	NA	R	NA
GSM366230.CEL	R	R	NA	R	NA	R	NA	NR	NA
GSM366231.CEL	R	R	NA	R	NA	R	NA	R	NA
GSM366232.CEL	R	R	NA	R	NA	R	NA	R	NA
GSM366233.CEL	R	NR	NR	R	R	R	R	R	R
GSM366234.CEL	R	R	NA	R	NA	R	NA	R	NA
GSM366235.CEL	R	R	NA	R	NA	R	NA	R	NA
GSM366236.CEL	R	R	NA	R	NA	R	NA	R	NA
GSM366237.CEL	R	R	NA	R	NA	R	NA	R	NA