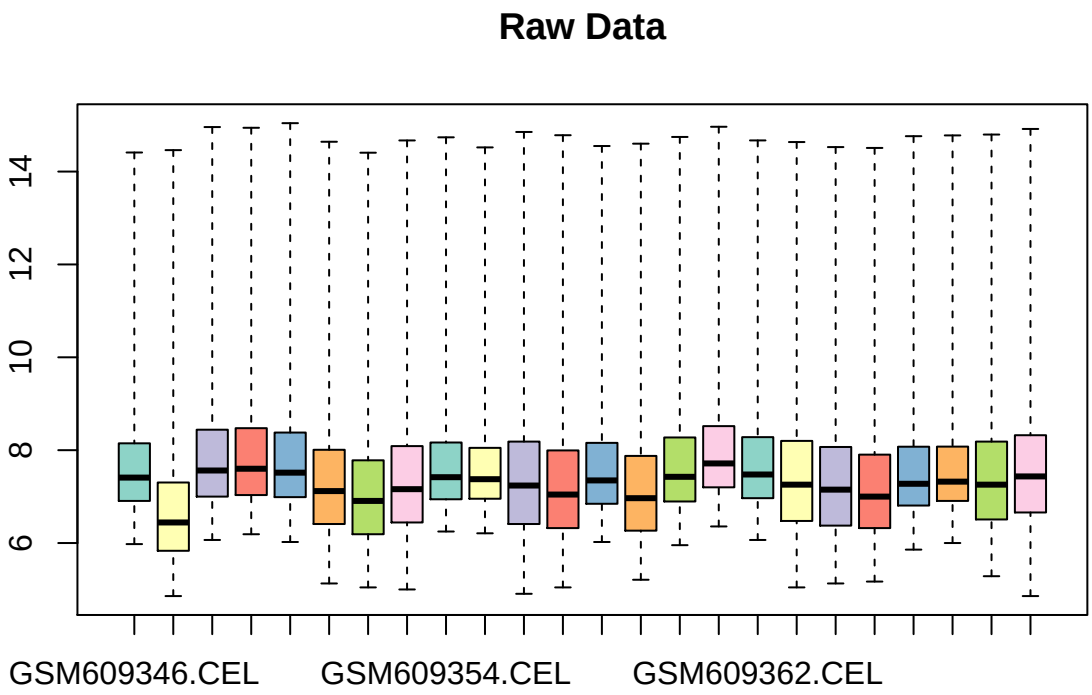
Lab 1

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

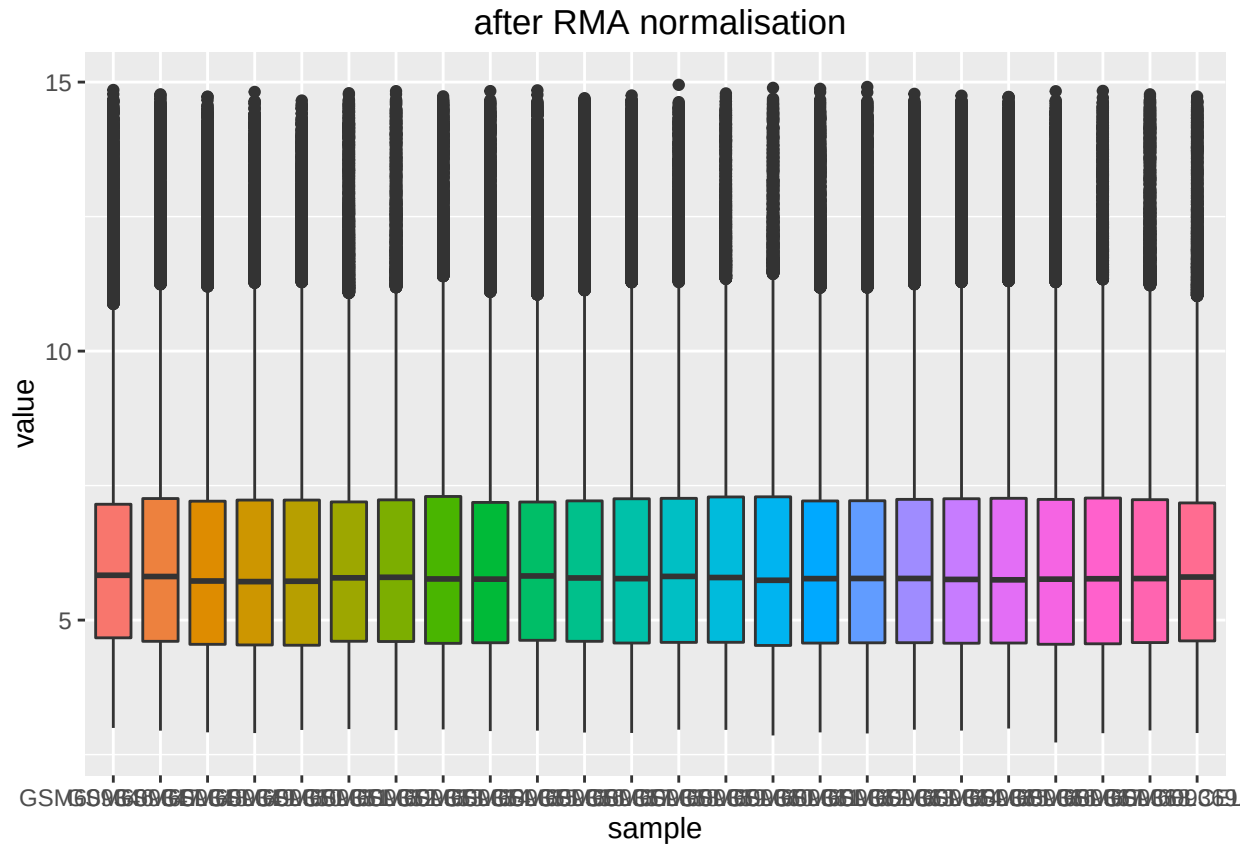
```
ggboxplot <-function ( dat, title){
  library(reshape2)
  plotData <- as.data.frame(t(exprs(dat)))
  plotData$sample <- row.names(plotData)
  plotData.m <- melt(plotData, id.vars = "sample")


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

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

Normailisation methods 1) RMA 2) Quantile only

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

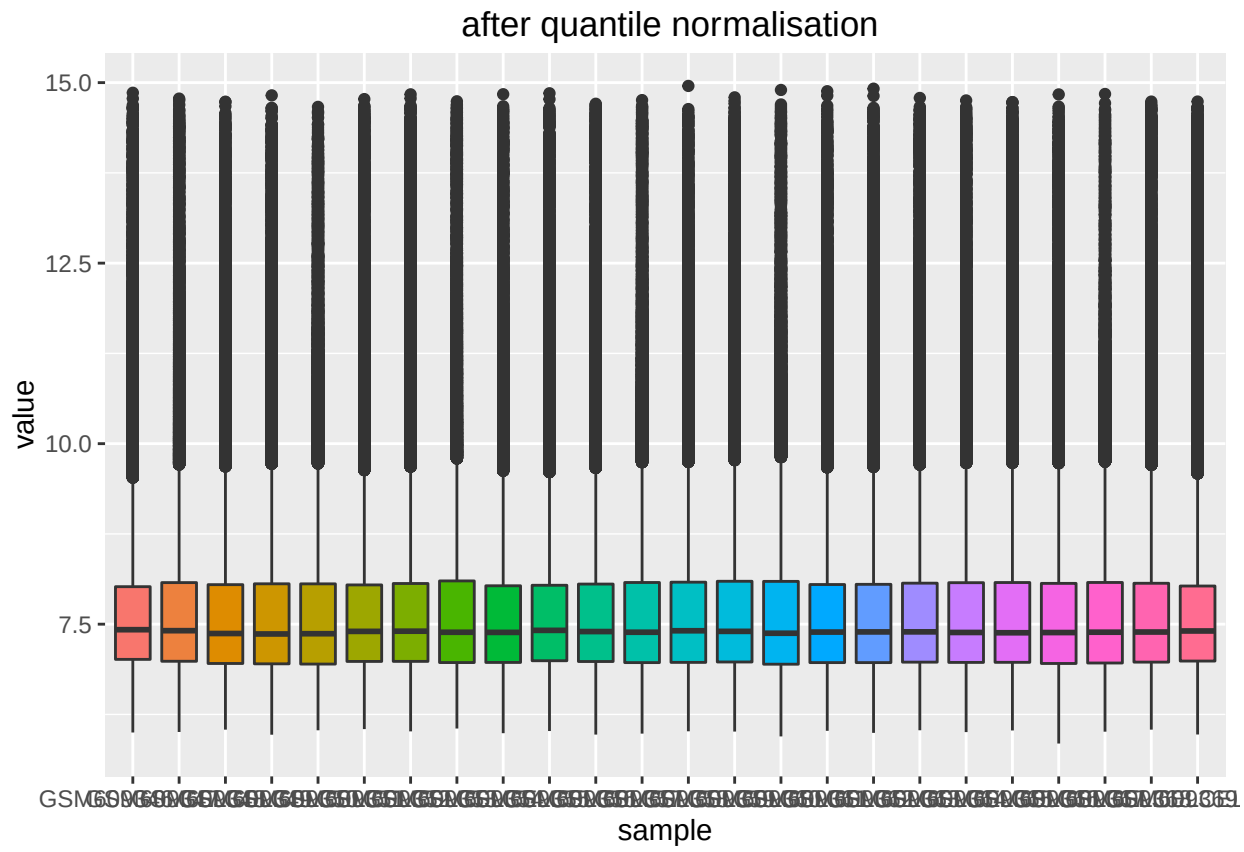
```
ggboxplot(eset, " after RMA normalisation")
```




```
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 inthere at the moment. sampleNames returns the names of the samples featureNames returns the probe IDs exprs returns the natrix of the expression values, probes X samples.

Add phenotype data to eset

Read in the phenotype data

```
pheno <- read.table("E-GEOD-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" ))
```

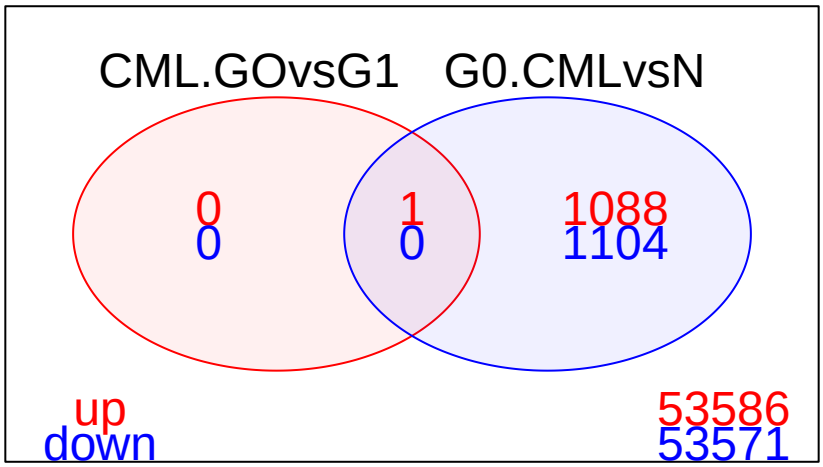
```
DC  <- paste(diseaseState, cellPhase, sep = ".")
DC <- factor(DC, levels = c("CML.G0" , "CML.G1", "N.G0", "N.G1"))
design <- model.matrix(~0+ DC)
colnames(design) <- levels(DC)
```

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.G0"   "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

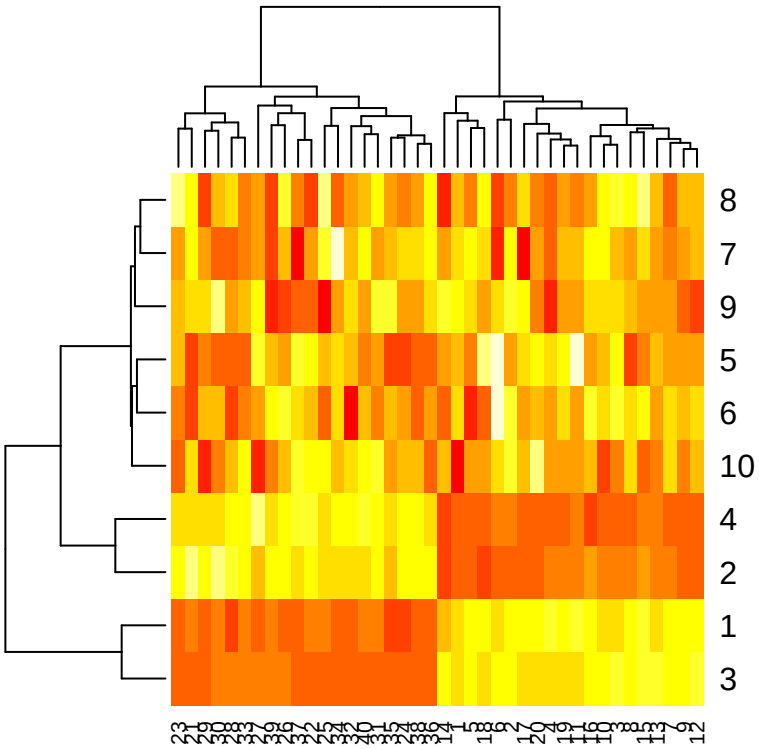
Labs 2 and 3: PCA and Clustering

2226768Y

9 March 2016

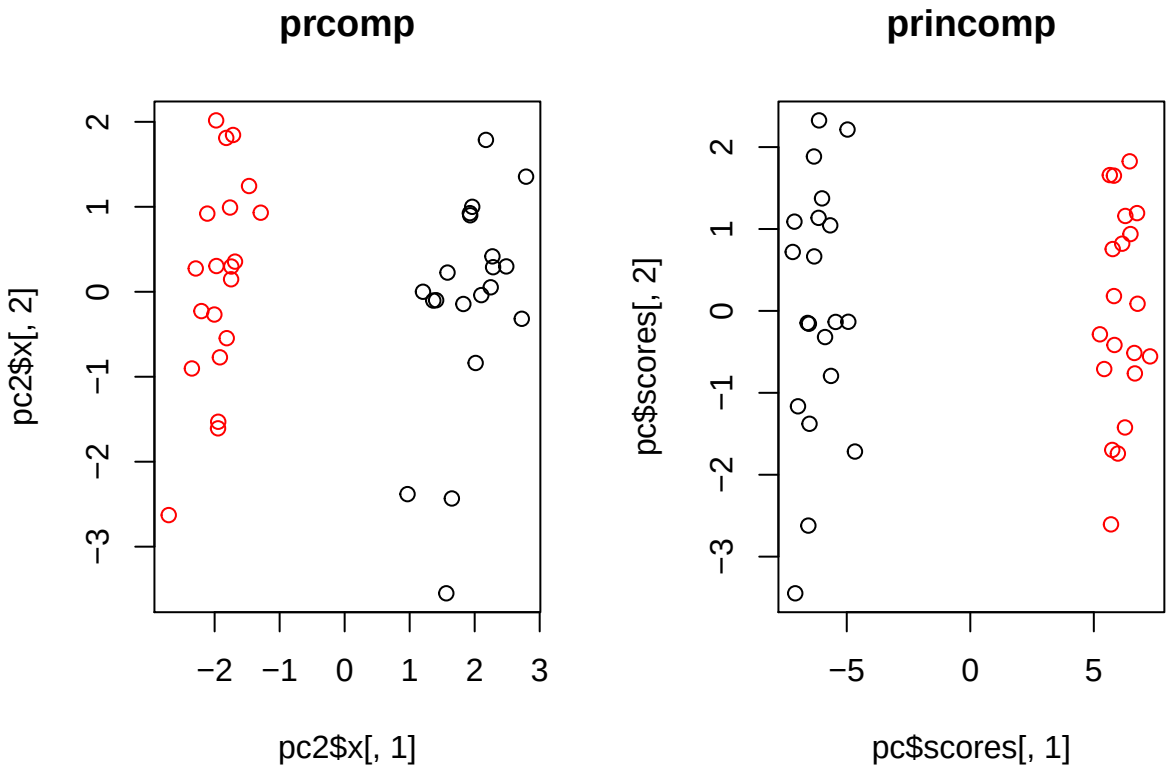
```
means <-c(3,-3,-2,2,2,-5,-2 ,5)
ma<- matrix(data=NA, 10,40)
set.seed(1)
for (i in 1:4){
  j<- 2*(i-1) +1
  ma[i,1:20] <- mvrnorm(20, means[j],1)
  ma [i,21:40] <- mvrnorm(20, means[j+1] ,1)
  }

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

1. Generating data Plot a Heatmap



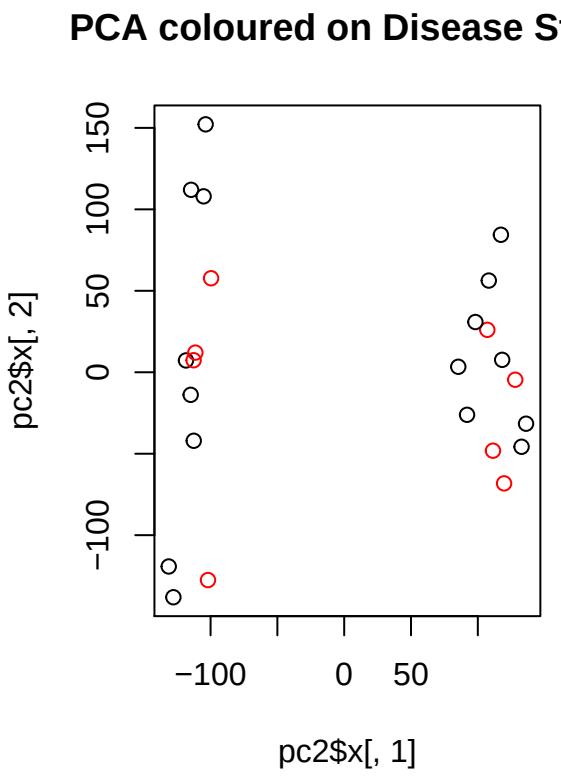
Plot PCA

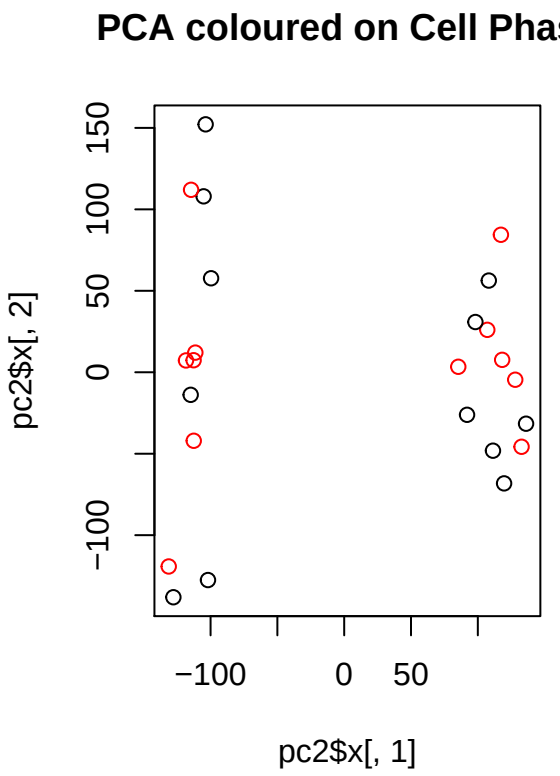
```
pc <- princomp(t(ma),scale. =T,center = T)
pc2 <- prcomp (t(ma),scale. =T,center = T, retx =T)
cond = c("healthy","disease")
metadata <- rep (cond,each = 20)
par(mfrow = c(1, 2))
plot (pc2$x[,1],pc2$x[,2],col = as.factor(metadata),main="prcomp" )
plot (pc$scores[,1],pc$scores[,2],col = as.factor(metadata),main="princomp")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



This shows very similar separation. Note scaling differance 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")
```





The data does not seperate on the phenotype of the data meaning the biggest changes in differentiation are no 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)
```

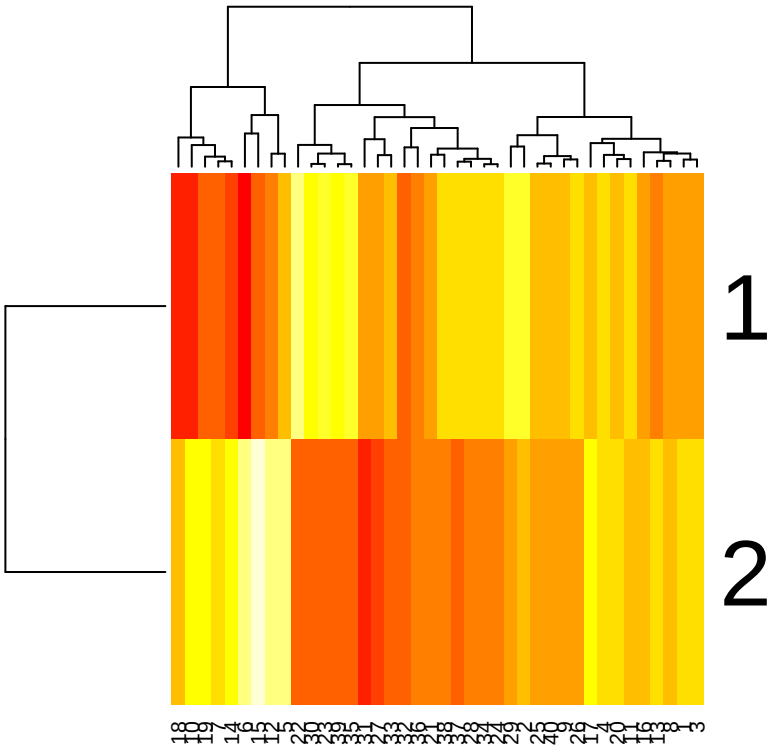
	sort.abs.pc2.rotation. . . 2. . . .decreasing. . . TRUE..1.10.
228418_at	0.0123467
213984_at	0.0121544
220199_s_at	0.0120594
209852_x_at	0.0119917
225926_at	0.0119758
203428_s_at	0.0118907
227108_at	0.0117744
222714_s_at	0.0117568
223077_at	0.0117036
228578_at	0.0116812

Lab 3

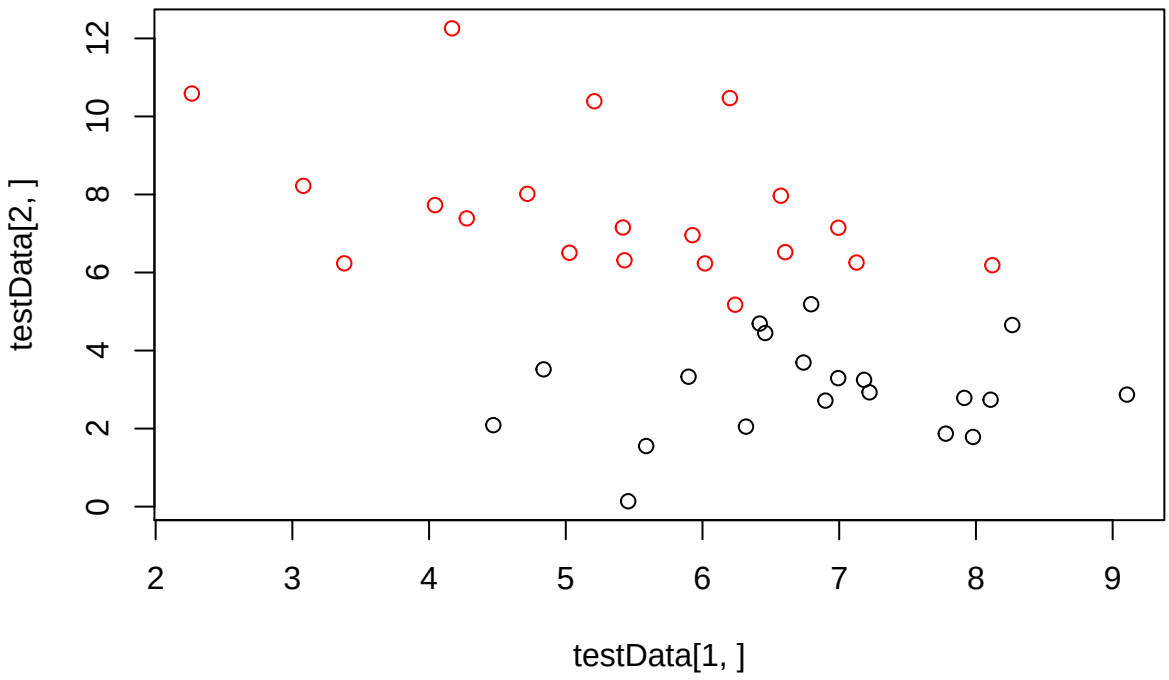
1. Hierarchical Clustering on Test Data

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)
```

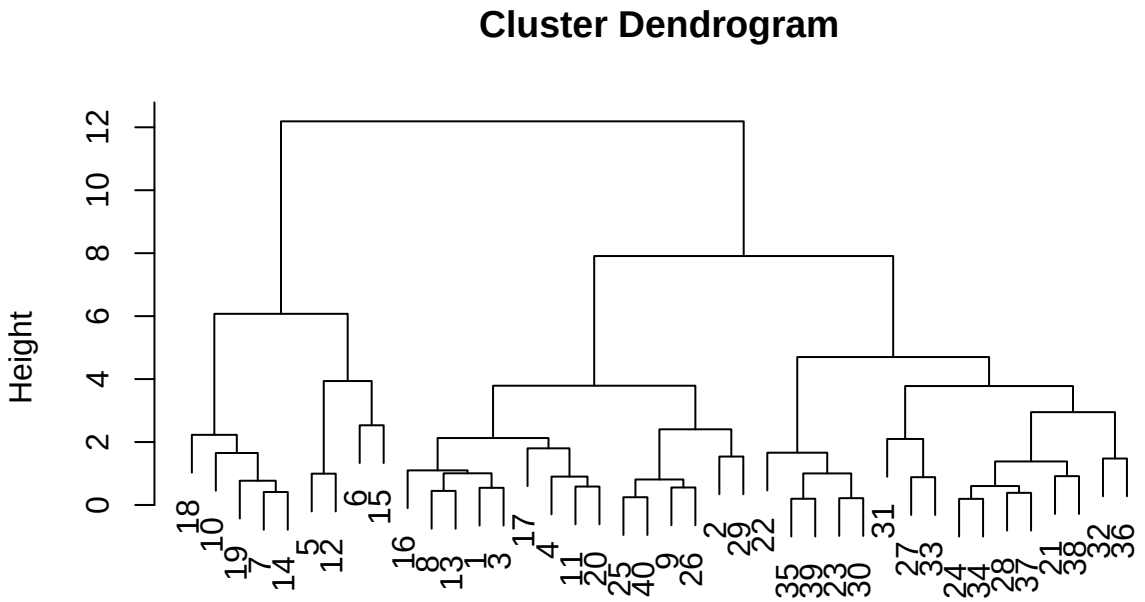



```
plot (testData[1,],testData[2,],col = as.factor(metadata) )
```

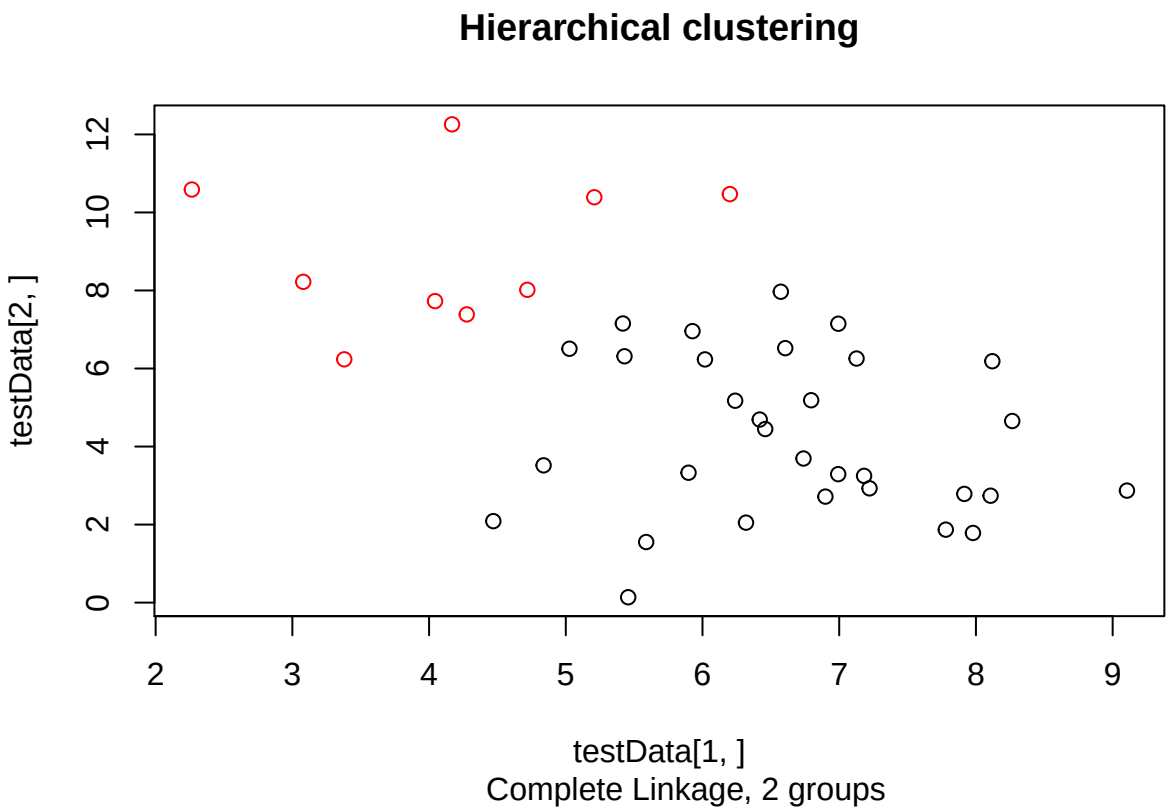


2) Hierarchical Clustering

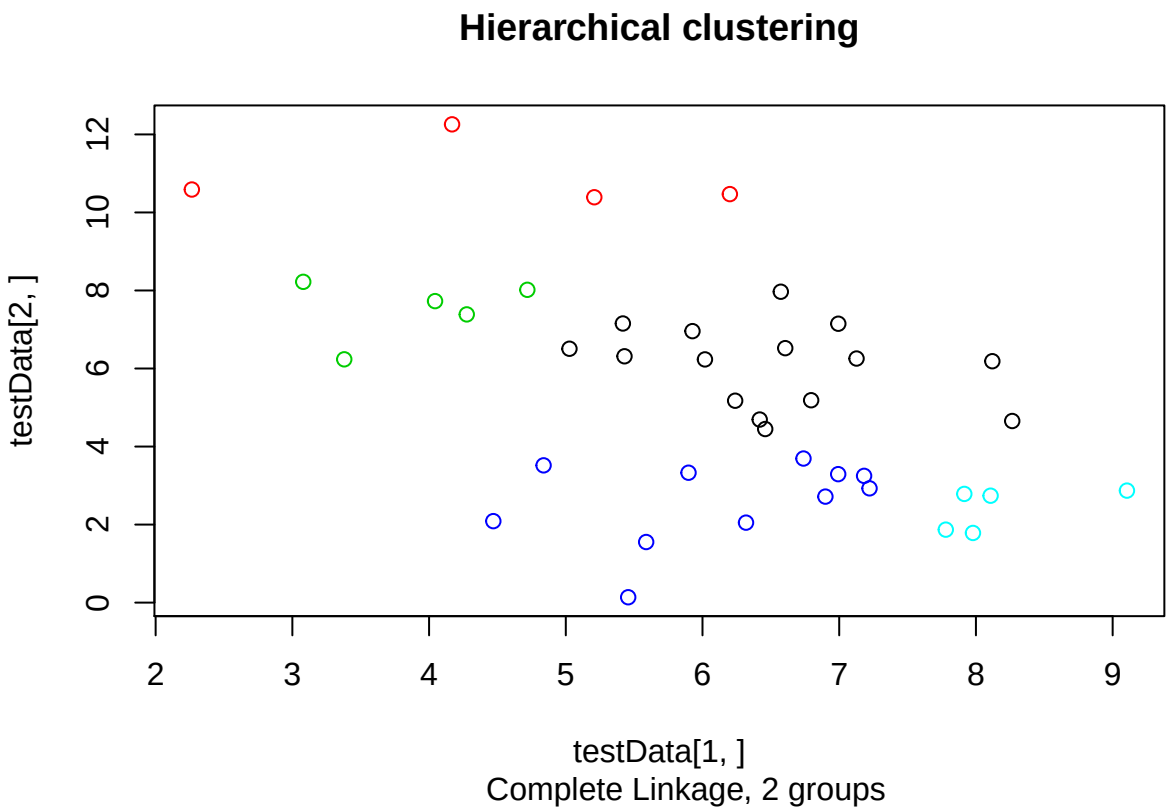
```
# Complete linkage
d = dist(t(testData))
hc = hclust(d,method="complete",members=NULL)
plot(hc)
```




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

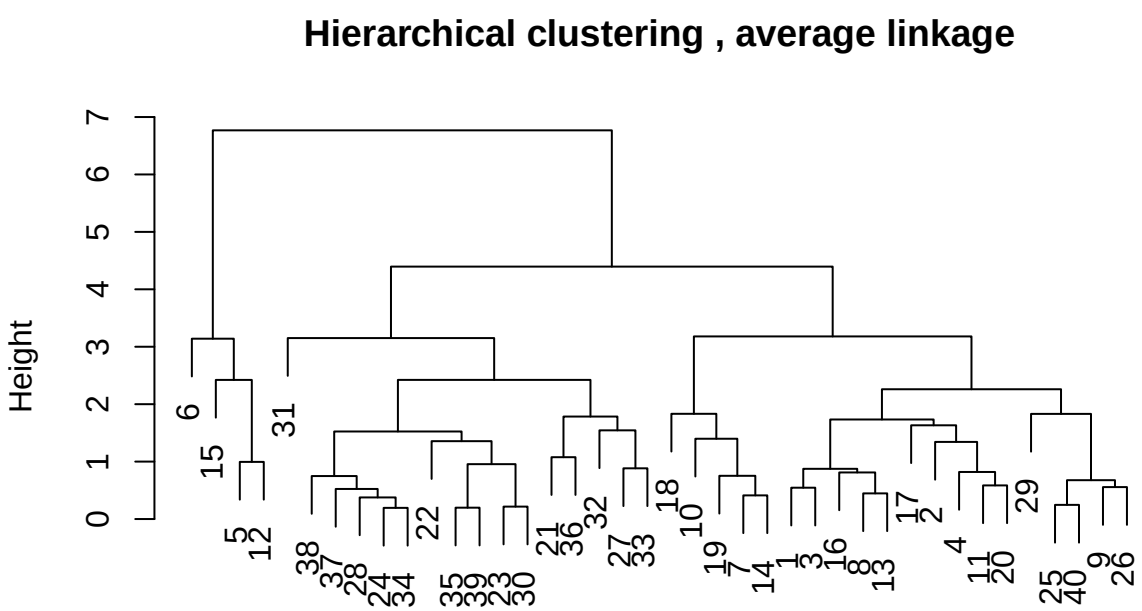



```
ct2 <- cutree(hc,5)
plot (testData[1,],testData[2,],col = ct2, main = "Hierarchical clustering" ,sub ="
      Complete Linkage, 2 groups" )
```

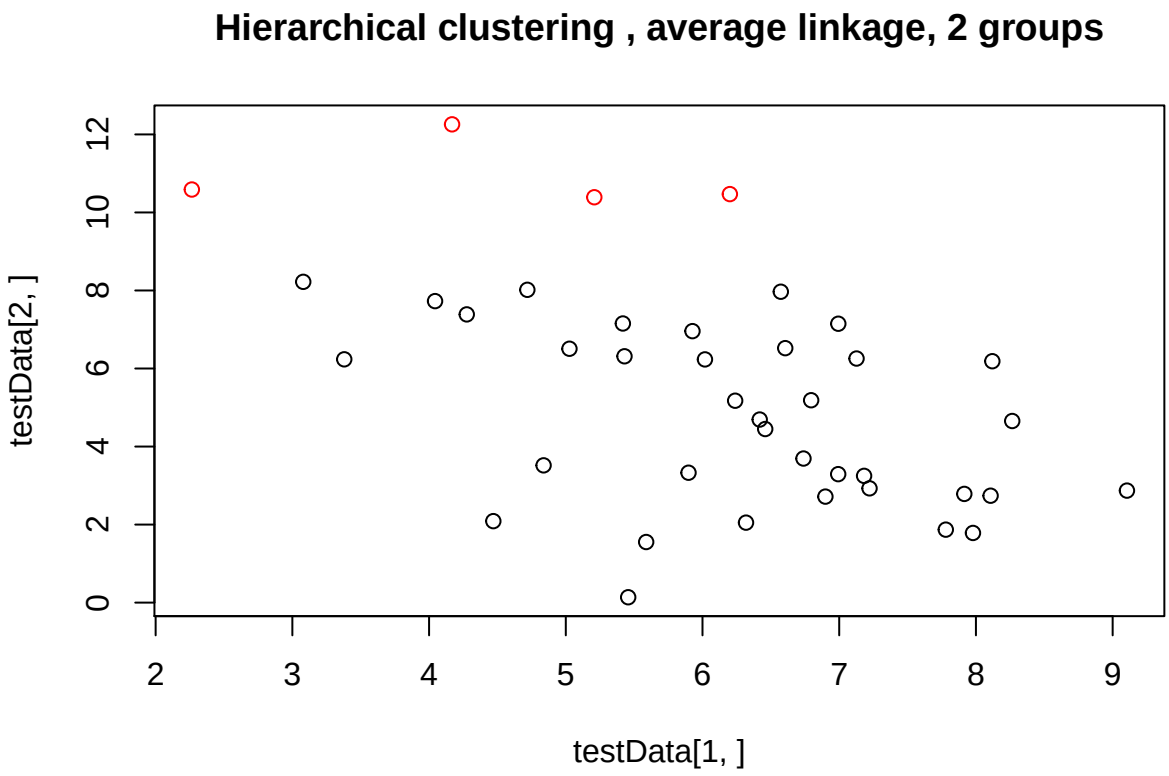


Average Linkage

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

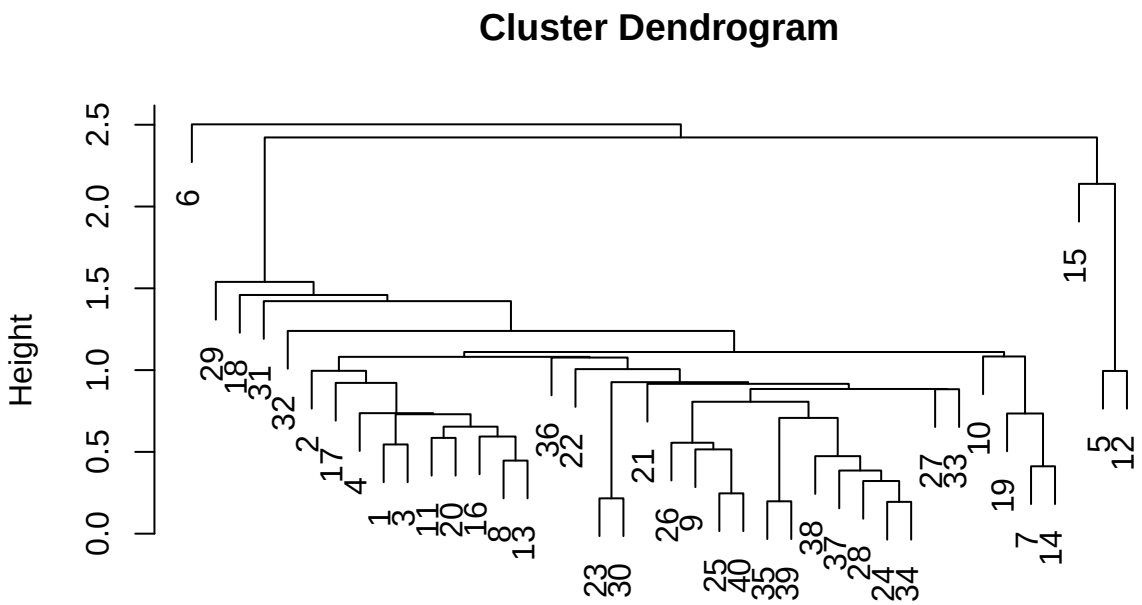



```
ct <-cutree (hca , k=2)

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

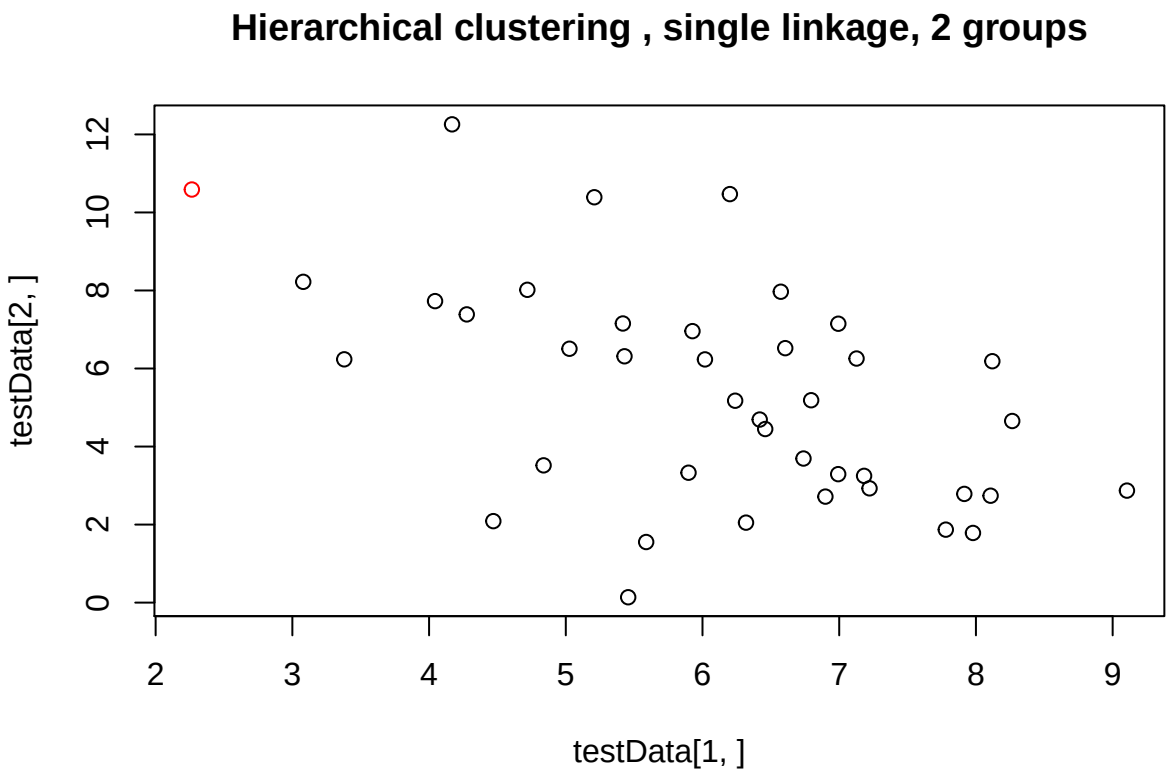


Single Linkage

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




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

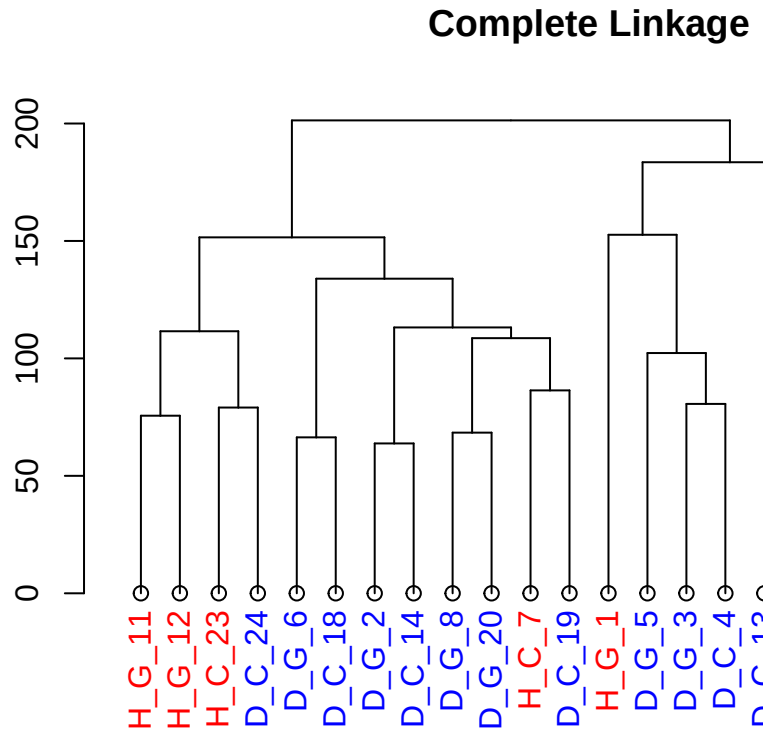
```
dis <- factor(phenoType$disease.state, labels= c("D","H"))
cell <- factor(phenoType$cell.phase , labels= c("G","C"))

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
```

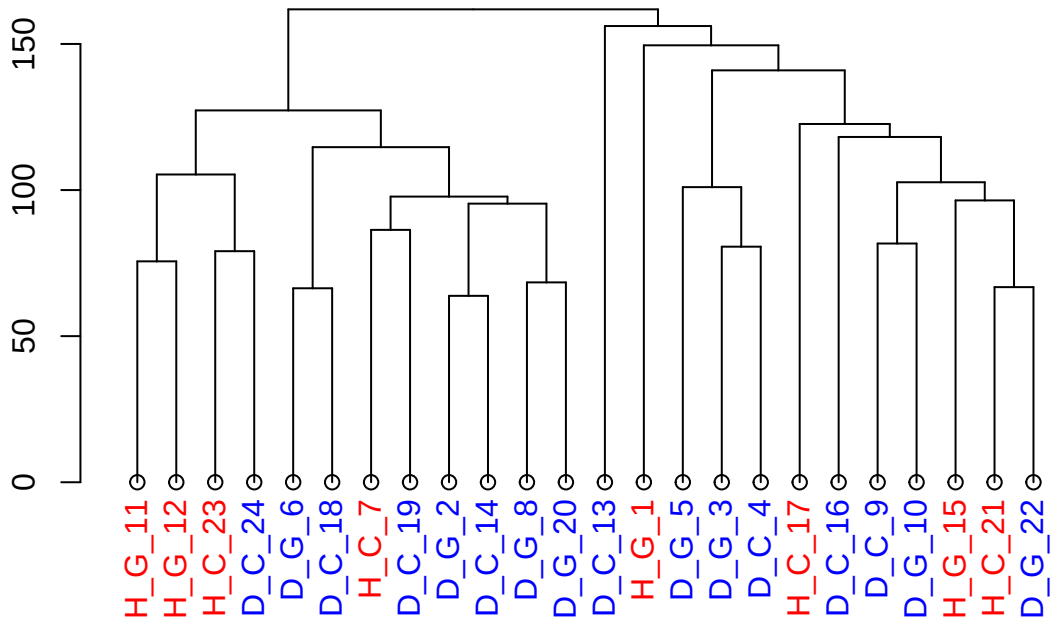
```
    attr(x, "nodePar") <- list(lab.col=colorCodes[code])
  }
  return(x)
}
```

```
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")
```

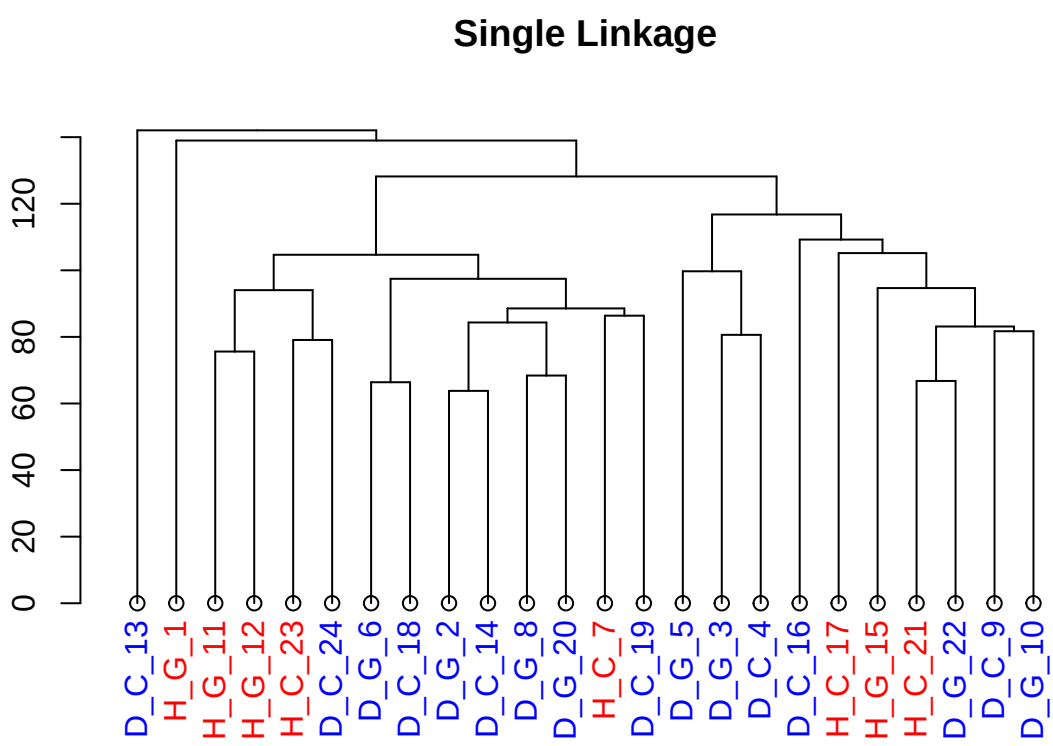


Differant Clustering methods for CML Data

```
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")
```




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

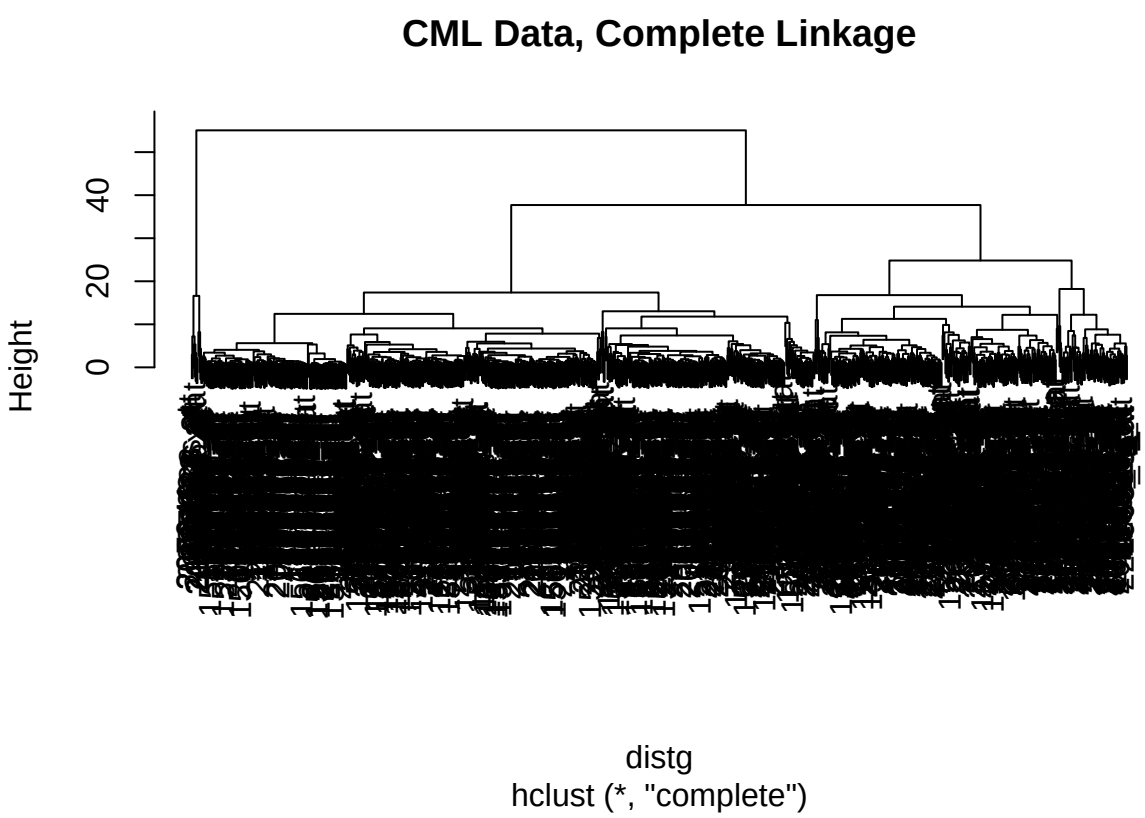


Q6) Clustering Genes Selectig a random subset of genes cluster to avoid computer crashes.

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

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

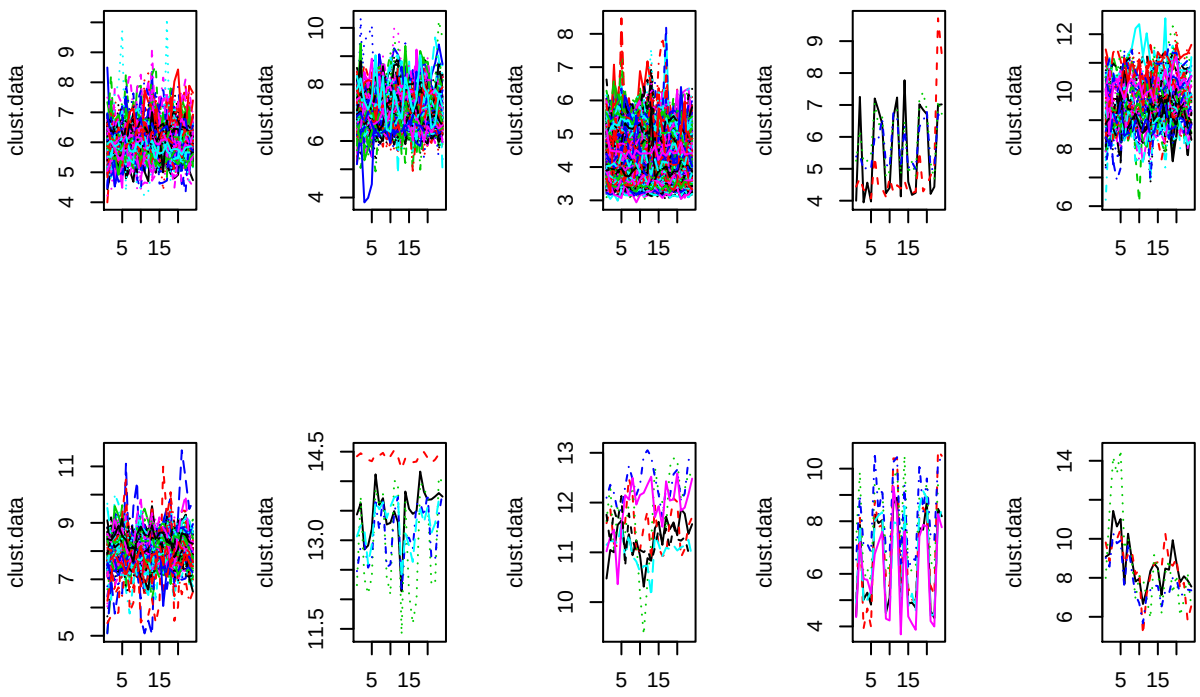
```
distg <- dist(data.genes1000)
hcma = hclust(distg,method="complete",members=NULL )

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




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

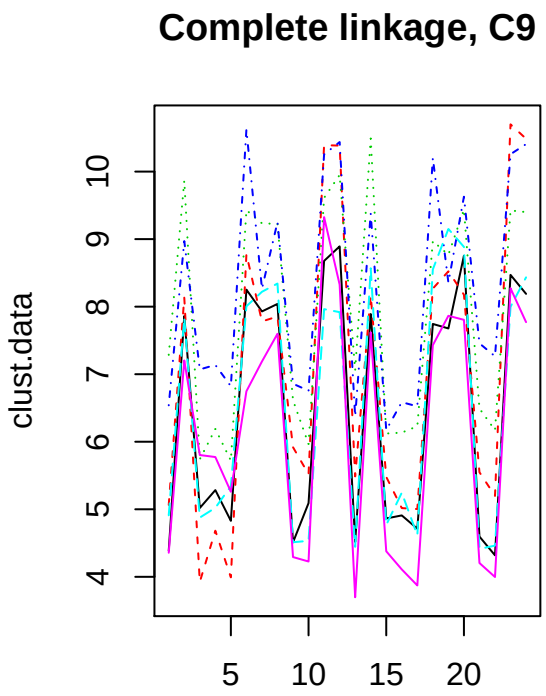
Plot all the clusters.

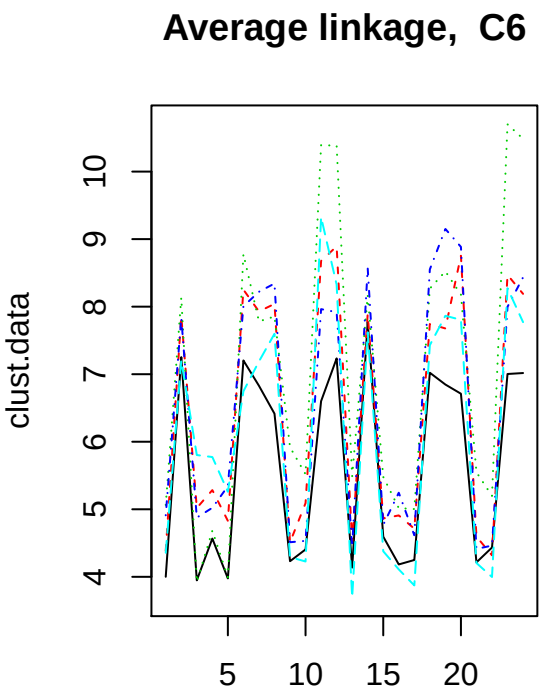
```
par(mfrow = c(2, 5))

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")
```





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)
```

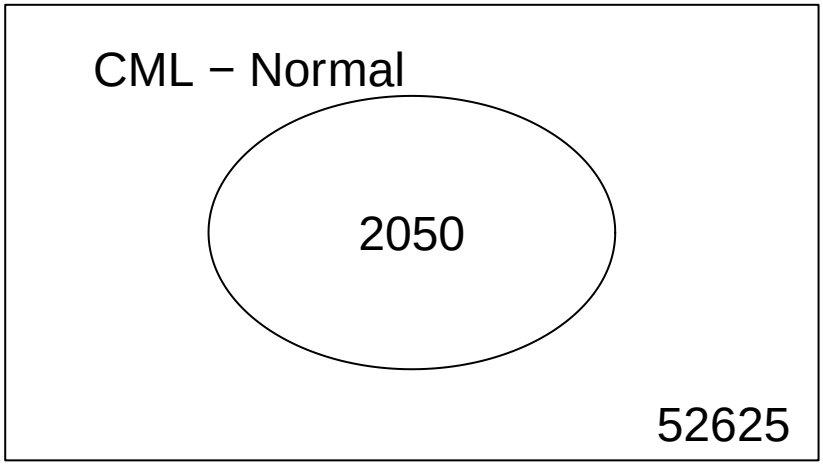
Read in the CEL files in the directory, then normalize the data

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

# reorder the phenoSubsetthe 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
```

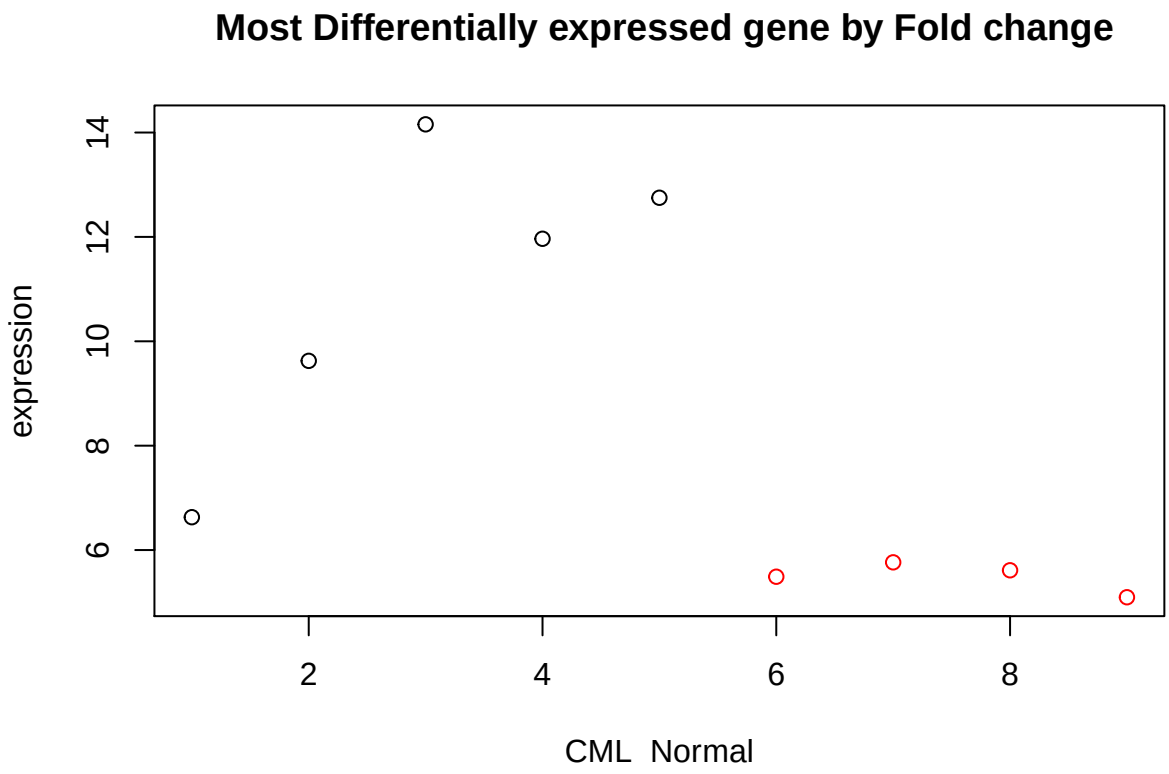
```
# create design array for all data where CML subset ==CP

goSamples <- pData(eset)$CMLphase != "AP" & pData(eset)$cellPhase =="GO"
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)
```

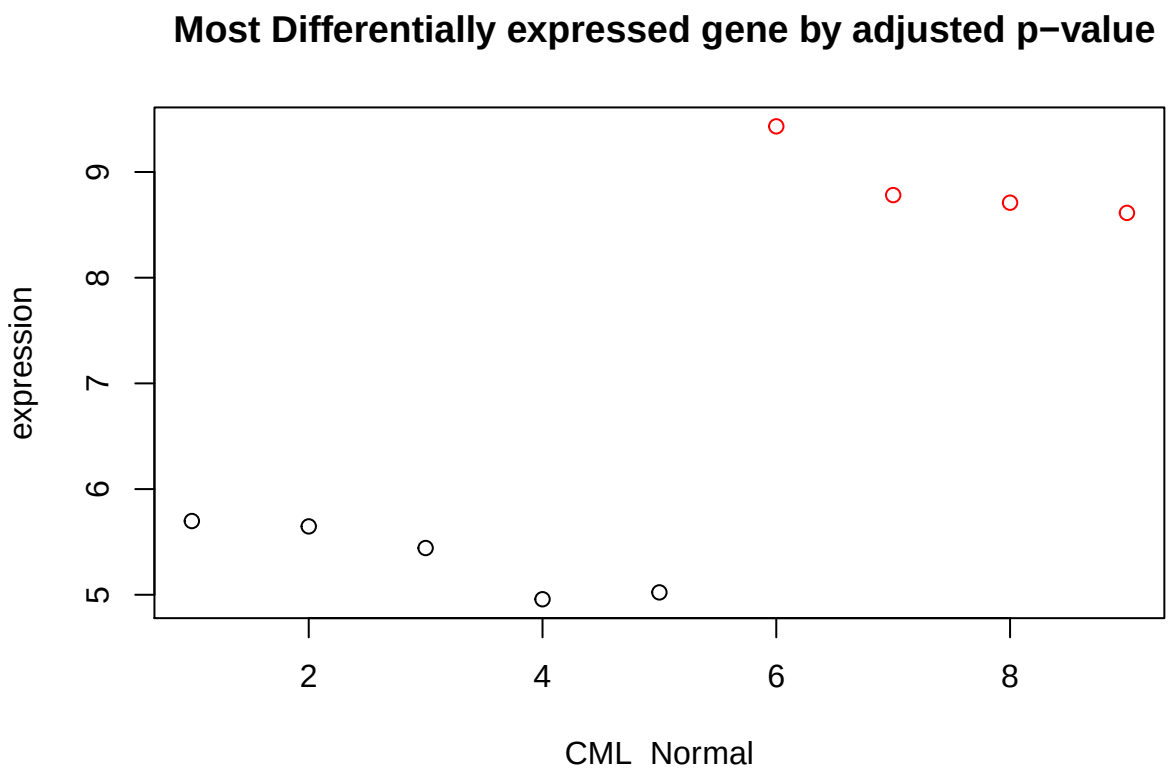


Analysing the data

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




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



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)
```

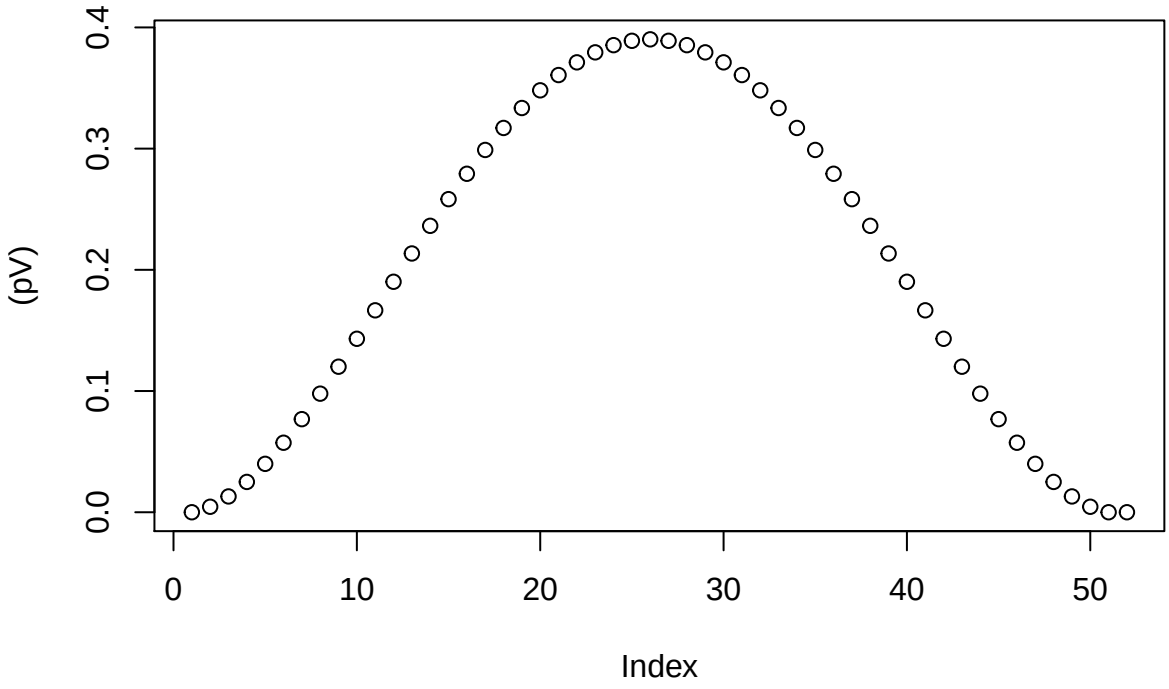
```
## [1] 0.0189012
```

```
#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))
```



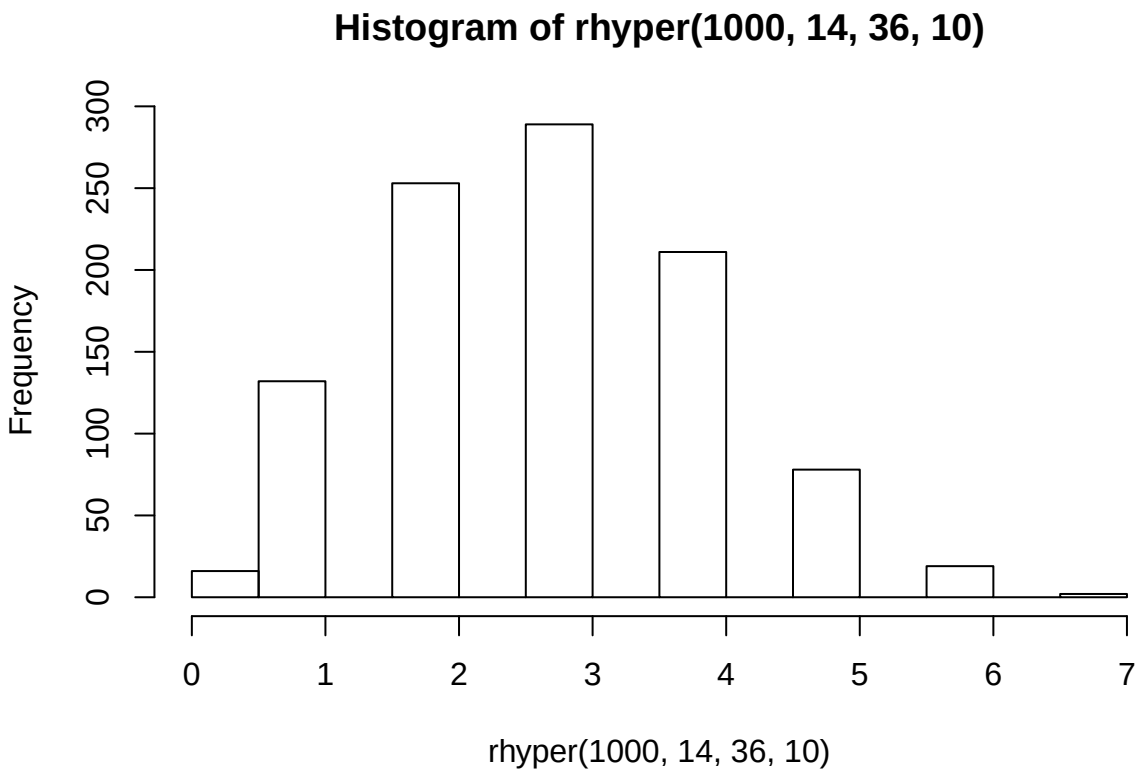

```
which (pV > .25)
```

```
##  [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 x (13-6) =14 (m), 52-14 =36 (n),drawn 10 times (k)

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



###3}Gene ontology

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

Use hyperGTest, first set up parameters