Genomics and Bioinformatics TCGA Project

Hamd Bilal Tahir

Professor Gang Fang

Introduction

Our main objective for this project was to make cancer gene co-expression modules for two cancer types in 10 patients (5 patients per cancer). For this, two cancer types were chosen: Breast Cancer (BRCA) and Ovarian Cancer (OV).

The second objective of our project was to identify the Point Mutation Frequencies for genes in each cancer type (BRCA and OV) and the Point Mutation Spectra for each cancer type.

Materials and Methods

We started off by opening the TCGA Website and launching the data portal. From there, we downloaded samples from the GDC (Genomic Data Commons) Portal fom the NIH (National Cancer Institute). The cancer types were searched and 5 transcriptome profiling patient files were downloaded each for both BRCA and OV. The files contained the "Gene ID" data and their FPKM values. The values indicated the fragments per kilobase of transcript found per a million mapped reads.

Then, we downloaded the Cosmic Census genes for both BRCA and OV in order to limit the number of genes to be worked on. The goal was to intersect the patient data with the Cosmic Census data in order to only obtain the FPKM values for Genes in the patient data that were found in the Cosmic Census Genes. But the Cosmic data had "Associated Gene Name" data whereas the TCGA patient files had "Gene ID" data. In order to link them together, we used Biomart from the Ensemble website. We put in the "Associated Gene

Name" data from the Cosmic Census Gene files as input into Biomart, and, as a result, obtained their corresponding "Gene ID" and "Transcription ID" data. The "Gene ID" data was sorted according to ascending order and the corresponding "Associated Gene Name" were sorted accordingly, in order to keep the correct structure of all the data.

Now, all the important data was to be obtained by coding in R. The following code imported all the necessary files and saved them into vectors.

```
#First Cancer
name='Breast Cancer' #The First Cancer
drctry=paste('E:/NYU/Fall 2016/Genomics and Bioinformatics/TCGA Project/Project/Combined Data/',name,sep='')
setwd(drctry) #set directory
name_B=paste(name,' Biomart','.csv',sep='')
name_C=paste(name,' Gene Names','.csv',sep='')
brca_vec_1=read.csv(file=name_B,as.is=T)         #Biomart Data
brca_vec_2=read.csv(file=name_C,as.is=T)         #Associated Gene Names Data
sample_1=read.csv(file='Sample 1.csv',as.is=T)    #Sample 1 BRCA
sample_2=read.csv(file='Sample 2.csv',as.is=T)
sample_3=read.csv(file='Sample 3.csv',as.is=T)
sample_4=read.csv(file='Sample 4.csv',as.is=T)
sample_5=read.csv(file='Sample 5.csv',as.is=T)
brca_biomart_gene.id=(brca_vec_1$Gene.ID[1:length(brca_vec_1$Gene.ID)])
brca_vec_cosmic=(brca_vec_2$Associated.Gene.Name[1:length(brca_vec_2$Associated.Gene.Name)])
tcga_sample_1_GID=(sample_1$Gene.ID[1:length(sample_1$Gene.ID)])
tcga_sample_1_val=(sample_1$Expression.Values[1:length(sample_1$Expression.Values)])
tcga_sample_2_GID=(sample_2$Gene.ID[1:length(sample_2$Gene.ID)])
tcga_sample_2_val=(sample_2$Expression.Values[1:length(sample_2$Expression.Values)])
tcga_sample_3_GID=(sample_3$Gene.ID[1:length(sample_3$Gene.ID)])
tcga_sample_3_val=(sample_3$Expression.Values[1:length(sample_3$Expression.Values)])
tcga_sample_4_GID=(sample_4$Gene.ID[1:length(sample_4$Gene.ID)])
tcga_sample_4_val=(sample_4$Expression.Values[1:length(sample_4$Expression.Values)])
tcga_sample_5_GID=(sample_5$Gene.ID[1:length(sample_5$Gene.ID)])
tcga_sample_5_val=(sample_5$Expression.Values[1:length(sample_5$Expression.Values)])
```

```
#2nd Cancer
name='Ovarian Cancer' #Second Cancer
drctry=paste('E:/NYU/Fall 2016/Genomics and Bioinformatics/TCGA Project/Project/Combined Data/',name,sep='')
setwd(drctry)
name_B=paste(name,' Biomart','.csv',sep='')
name_C=paste(name,' Gene Names','.csv',sep='')
ov_vec_1=read.csv(file=name_B,as.is=T)
ov_vec_2=read.csv(file=name_C,as.is=T)
sample_6=read.csv(file='Sample 1.csv',as.is=T)
sample_7=read.csv(file='Sample 2.csv',as.is=T)
sample_8=read.csv(file='Sample 3.csv',as.is=T)
sample_9=read.csv(file='Sample 4.csv',as.is=T)
sample_10=read.csv(file='Sample 5.csv',as.is=T)
ov_biomart_gene.id=(ov_vec_1$Gene.ID[1:length(ov_vec_1$Gene.ID)])
ov_vec_cosmic=(ov_vec_2$Associated.Gene.Name[1:length(ov_vec_2$Associated.Gene.Name)])
tcga_sample_6_GID=(sample_6$Gene.ID[1:length(sample_6$Gene.ID)])
tcga_sample_6_val=(sample_6$Expression.Values[1:length(sample_6$Expression.Values)])
tcga_sample_7_GID=(sample_7$Gene.ID[1:length(sample_7$Gene.ID)])
tcga_sample_7_val=(sample_7$Expression.Values[1:length(sample_7$Expression.Values)])
tcga_sample_8_GID=(sample_8$Gene.ID[1:length(sample_8$Gene.ID)])
tcga_sample_8_val=(sample_8$Expression.Values[1:length(sample_8$Expression.Values)])
tcga_sample_9_GID=(sample_9$Gene.ID[1:length(sample_9$Gene.ID)])
tcga_sample_9_val=(sample_9$Expression.Values[1:length(sample_9$Expression.Values)])
tcga_sample_10_GID=(sample_10$Gene.ID[1:length(sample_10$Gene.ID)])
tcga_sample_10_val=(sample_10$Expression.Values[1:length(sample_10$Expression.Values)])
```

Our Biomart data had multiple rows for the same Gene ID (primarily because of each Gene ID having multiple Transcription IDs). So we had to parse the file in a way that we only take one all Gene IDs once. The function used is as follows.

```
gene_id=function(vector){
 c1='nothing'
 new_vector=c()
```

```
 for (i in (1:length(vector))){
  c2=vector[i]
  if (c1!=c2){
   new_vector=c(new_vector,c2)
  }
  c1=c2
 }
 return (new_vector)
}
#First Cancer
brca_biomart_gene.id_final=gene_id(brca_biomart_gene.id)
#Second Cancer
ov_biomart_gene.id_final=gene_id(ov_biomart_gene.id)
```

The following function changed the “Gene ID” data for each patient into the appropriate “Gene ID” data (without the decimal values at the end of each Gene ID, such as “.19”). This was done so that a link could be made between the Cosmic data and the TCGA data.

```
#First Cancer
extract_GID=function(vector){
 new_vector=c()
 for (i in (1:length(vector))){
  x=gsub("[.+]",' ',vector[i])
  x=unlist(strsplit(x, " "))
  x=x[1]
  new_vector=c(new_vector,x)
 }
 return (new_vector)
}
tcga_sample_1_GID_new=extract_GID(tcga_sample_1_GID)
tcga_sample_2_GID_new=extract_GID(tcga_sample_2_GID)
tcga_sample_3_GID_new=extract_GID(tcga_sample_3_GID)
tcga_sample_4_GID_new=extract_GID(tcga_sample_4_GID)
```

```
tcga_sample_5_GID_new=extract_GID(tcga_sample_5_GID)


#2nd Cancer

tcga_sample_6_GID_new=extract_GID(tcga_sample_6_GID)

tcga_sample_7_GID_new=extract_GID(tcga_sample_7_GID)

tcga_sample_8_GID_new=extract_GID(tcga_sample_8_GID)

tcga_sample_9_GID_new=extract_GID(tcga_sample_9_GID)

tcga_sample_10_GID_new=extract_GID(tcga_sample_10_GID)
```

After we got the patient data for all 10 patients and the appropriate data from the Biomart Gene ID File, we merged the required Genes and their corresponding data for both BRCA and OV in the following code.

```
union_genes=function(vector1,vector2){                    #Function to find the union for two vectors
 x=c()
 for(i in (1:length(vector1))){
   x=c(x,vector1[i])
 }
 for(i in (1:length(vector2))){
   if (vector2[i] %in% vector1){
   }
   else{
     x=c(x,vector2[i])
   }
 }
 return(x)
}
drctry=paste('E:/NYU/Fall 2016/Genomics and Bioinformatics/TCGA Project/Project/Combined Data/',sep='')
setwd(drctry)  #new directory
```

```
total_genes_ID=union_genes(brca_biomart_gene.id_final,ov_biomart_gene.id_final)   #Union Gene IDs for BRCA and OV

total_genes_names=union_genes(brca_vec_cosmic,ov_vec_cosmic)                    #Union Associated Gene Names for BRCA and OV


extract_gene_val=function(vector1,vector2,vector3){      #function for extracting the appropriate FPKM values correspondingly

 new_vector=c()

 for (i in (1:length(vector1))){

  pos = which(vector2 %in% vector1[i])

  if (identical(pos,integer(0))){

   new_vector[i]=as.character(0)

  }

  else{

   new_vector[i]=as.character(vector3[pos])

  }

 }

 new_vector[is.na(new_vector)]='0'


 return (new_vector)

}

sample_1_sim_val_final=extract_gene_val(total_genes_ID,tcga_sample_1_GID_new,tcga_sample_1_val) #Patient 1 FPKM val

sample_2_sim_val_final=extract_gene_val(total_genes_ID,tcga_sample_2_GID_new,tcga_sample_2_val)

sample_3_sim_val_final=extract_gene_val(total_genes_ID,tcga_sample_3_GID_new,tcga_sample_3_val)

sample_4_sim_val_final=extract_gene_val(total_genes_ID,tcga_sample_4_GID_new,tcga_sample_4_val)

sample_5_sim_val_final=extract_gene_val(total_genes_ID,tcga_sample_5_GID_new,tcga_sample_5_val)

sample_6_sim_val_final=extract_gene_val(total_genes_ID,tcga_sample_6_GID_new,tcga_sample_6_val)

sample_7_sim_val_final=extract_gene_val(total_genes_ID,tcga_sample_7_GID_new,tcga_sample_7_val)

sample_8_sim_val_final=extract_gene_val(total_genes_ID,tcga_sample_8_GID_new,tcga_sample_8_val)

sample_9_sim_val_final=extract_gene_val(total_genes_ID,tcga_sample_9_GID_new,tcga_sample_9_val)

sample_10_sim_val_final=extract_gene_val(total_genes_ID,tcga_sample_10_GID_new,tcga_sample_10_val)
```

Finally, all the data was saved in a big matrix and written to a .csv Excel file.

```
#Making matrices for each patient data
matrix_1=matrix(total_genes_names,nrow=length(total_genes_names),byrow=T)
matrix_2=matrix(total_genes_ID,nrow=length(total_genes_ID),byrow=T)
matrix_3=matrix(0,nrow=1,ncol=12)
matrix_sample_1=matrix(as.numeric(sample_1_sim_val_final),nrow=length(sample_1_sim_val_final),byrow=T)
matrix_sample_2=matrix(as.numeric(sample_2_sim_val_final),nrow=length(sample_2_sim_val_final),byrow=T)
matrix_sample_3=matrix(as.numeric(sample_3_sim_val_final),nrow=length(sample_3_sim_val_final),byrow=T)
matrix_sample_4=matrix(as.numeric(sample_4_sim_val_final),nrow=length(sample_4_sim_val_final),byrow=T)
matrix_sample_5=matrix(as.numeric(sample_5_sim_val_final),nrow=length(sample_5_sim_val_final),byrow=T)
matrix_sample_6=matrix(as.numeric(sample_6_sim_val_final),nrow=length(sample_6_sim_val_final),byrow=T)
matrix_sample_7=matrix(as.numeric(sample_7_sim_val_final),nrow=length(sample_7_sim_val_final),byrow=T)
matrix_sample_8=matrix(as.numeric(sample_8_sim_val_final),nrow=length(sample_8_sim_val_final),byrow=T)
matrix_sample_9=matrix(as.numeric(sample_9_sim_val_final),nrow=length(sample_9_sim_val_final),byrow=T)
matrix_sample_10=matrix(as.numeric(sample_10_sim_val_final),nrow=length(sample_10_sim_val_final),byrow=T)
matrix_3[1,1]='Gene Name'
matrix_3[1,2]='Gene ID'
for(i in (1:10)){
  matrix_3[1,i+2]=paste('Patient',as.character(i),sep=' ')
}
#Binding all the matrices
final_matrix=cbind(matrix_1,matrix_2)
final_matrix=cbind(final_matrix,as.numeric(matrix_sample_1))
final_matrix=cbind(final_matrix,as.numeric(matrix_sample_2))
final_matrix=cbind(final_matrix,as.numeric(matrix_sample_3))
final_matrix=cbind(final_matrix,as.numeric(matrix_sample_4))
final_matrix=cbind(final_matrix,as.numeric(matrix_sample_5))
final_matrix=cbind(final_matrix,as.numeric(matrix_sample_6))
final_matrix=cbind(final_matrix,as.numeric(matrix_sample_7))
final_matrix=cbind(final_matrix,as.numeric(matrix_sample_8))
final_matrix=cbind(final_matrix,as.numeric(matrix_sample_9))
final_matrix=cbind(final_matrix,as.numeric(matrix_sample_10))
final_matrix=rbind(matrix_3,final_matrix)
```

```
write.table(final_matrix,file='Total Gene Expressions (BRCA and OV).csv',sep=',',row.names = F, col.names=F,append=FALSE)
```

WGCNA Analysis

After we got our needed data, we used the WGCNA package to analyze our data (Code not Shown). For the WGCNA, in order to correctly use the "simulateDatExpr5Modules" function (the function that simulated data expression modules for our data) we had to take out 2 low LQ-FPKM genes so that we had a sample of 53 genes. We put our no. of observations=10 (the number of our patients) and no. of genes=53. Then we imported our main .csv Excel data file along with the tutorial file from WGCNA: datTraits file (this tutorial file was used as the standard clinical trait according to which our data was to be analyzed). After this, we did further analysis which is shown in the "Results" section.

Mutation Frequency and Spectra Analysis

We downloaded the big data from the "Muse" pipeline for both BRCA and OV. We worked on MAF files that had mutation data for all the samples from the "Muse" pipeline. The data we obtained had many genes mutations for many samples. There were instances when the same gene ID corresponded to a different position in the genome (hence a different mutation). We parsed the data in a way that we only get one Gene ID per one gene but keep track of all the different locations and instances when the associated mutations happened, hence finding out the mutation frequency for the gene. Furthermore, we had to keep track of what all these mutations (at the different locations for the same gene) were (i.e C>T or A>G? e.t.c), so that we could make our mutation spectra.

To begin, we retrieved all the different genes (by retrieving their "Gene ID" data from the big data for each cancer). The following code represents the logic. (For keeping it short, only the code for one type of cancer is shown).

```
setwd('E:/NYU/Fall 2016/Genomics and Bioinformatics/TCGA Project/Project')  #Setting directory
library(TCGAbiolinks)
```

```
BRCA.muse.maf <- GDCquery_Maf("BRCA", pipelines = "muse")      #downloading BRCA data from pipeline
pos=c()
mutations=c()
data=c()
done_pos=c()
gene_ID=c()
for (i in 1:length(BRCA.muse.maf)){             #for loop for finding out the Gene IDs and number of mutations for each gene
 if(!(i%in%done_pos)){
  pos=which(BRCA.muse.maf$Gene%in%BRCA.muse.maf$Gene[i])
  mutations=c(mutations,length(pos))
  done_pos=c(done_pos,pos)
  data=c(BRCA.muse.maf[pos,])
  gene_ID=c(gene_ID,BRCA.muse.maf$Gene[i])
 }
 pos=c()
 data=c()
}

m1=matrix(gene_ID)                              #matrix of gene IDs
m2=matrix(mutations)                            #matrix of number of mutations corresponding to each gene ID in matrix m1
```

After this, we got the "Associated Gene Name" data from biomart for the genes in the matrix m1, and imported it into our environment. Then, we collected all the data in a form of a matrix.

```
vec=read.csv(file='Biomart for mutations (BRCA).csv',as.is=T)          #Read biomart file for BRCA
v1_biomart=vec$Gene.ID
v2_biomart=vec$Associated.Gene.Name
gene_name=c()
```

```
for (i in 1:length(gene_ID)){              #for loop for linking biomart data to Gene ID data for accurate allocation of data
 pos=which(v1_biomart%in%gene_ID[i])
 gene_name=c(gene_name,v2_biomart[pos])
}
m3=matrix(gene_name)                       #matrix of gene names
f_matrix=c()
f_matrix=cbind(m3,m1,m2)
m6=matrix(data=0,ncol=3)
f_matrix=rbind(m6,f_matrix)                #Matrix containing the Gene name, Gene ID and No. of mutations per gene
f_matrix[1,1]='Gene Name'
f_matrix[1,2]='Gene ID'
f_matrix[1,3]='Frequency of Somatic Mutations per Gene'
```

At the end, the mutation frequency data was saved in the form of a bar plot in a .pdf file (In the supplemental data) and as a data frame in an .csv Excel file.

```
write.table(f_matrix,file='Frequency of Point Mutations per Gene (BRCA).csv',col.names=F,row.names=F,sep=',')
pdf('Point Mutation Frequencies (BRCA).pdf',width=200,height=50)
mids=barplot(mutations,col=c('red'),main="Point Mutation Frequencies (BRCA)", cex.axis=5,  ylim=c(0,200),ylab='Frequency [%]')
axis(1, at=mids, labels=gene_name,las=1,cex.axis=1.5)
dev.off()
```

Finally, we parsed the data again to find out the types of substitutions (C>T or A>G? e.t.c) for each mutation. The code is shown below.

```
#Get number of substitutions for each substitution type

AtoG=0
for (i in 1:length(BRCA.muse.maf$Tumor_Seq_Allele1)){   #for loop to find the number of A>G substitutions
 if(!(BRCA.muse.maf$Tumor_Seq_Allele1[i]!='A') & !(BRCA.muse.maf$Tumor_Seq_Allele2[i]!='G')){
   AtoG=AtoG+1                  #No. of A>G substitutions
 }
}

AtoC=0
for (i in 1:length(BRCA.muse.maf$Tumor_Seq_Allele1)){
 if(!(BRCA.muse.maf$Tumor_Seq_Allele1[i]!='A') & !(BRCA.muse.maf$Tumor_Seq_Allele2[i]!='C')){
   AtoC=AtoC+1
 }
}

AtoT=0
for (i in 1:length(BRCA.muse.maf$Tumor_Seq_Allele1)){
 if(!(BRCA.muse.maf$Tumor_Seq_Allele1[i]!='A') & !(BRCA.muse.maf$Tumor_Seq_Allele2[i]!='T')){
   AtoT=AtoT+1
 }
}

GtoA=0
for (i in 1:length(BRCA.muse.maf$Tumor_Seq_Allele1)){
 if(!(BRCA.muse.maf$Tumor_Seq_Allele1[i]!='G') & !(BRCA.muse.maf$Tumor_Seq_Allele2[i]!='A')){
   GtoA=GtoA+1
 }
}
GtoC=0
for (i in 1:length(BRCA.muse.maf$Tumor_Seq_Allele1)){
 if(!(BRCA.muse.maf$Tumor_Seq_Allele1[i]!='G') & !(BRCA.muse.maf$Tumor_Seq_Allele2[i]!='C')){
```

```
   GtoC=GtoC+1
  }
}
GtoT=0
for (i in 1:length(BRCA.muse.maf$Tumor_Seq_Allele1)){
  if(!(BRCA.muse.maf$Tumor_Seq_Allele1[i]!='G') & !(BRCA.muse.maf$Tumor_Seq_Allele2[i]!='T')){
    GtoT=GtoT+1
  }
}

CtoA=0
for (i in 1:length(BRCA.muse.maf$Tumor_Seq_Allele1)){
  if(!(BRCA.muse.maf$Tumor_Seq_Allele1[i]!='C') & !(BRCA.muse.maf$Tumor_Seq_Allele2[i]!='A')){
    CtoA=CtoA+1
  }
}

CtoG=0
for (i in 1:length(BRCA.muse.maf$Tumor_Seq_Allele1)){
  if(!(BRCA.muse.maf$Tumor_Seq_Allele1[i]!='C') & !(BRCA.muse.maf$Tumor_Seq_Allele2[i]!='G')){
    CtoG=CtoG+1
  }
}

CtoT=0
for (i in 1:length(BRCA.muse.maf$Tumor_Seq_Allele1)){
  if(!(BRCA.muse.maf$Tumor_Seq_Allele1[i]!='C') & !(BRCA.muse.maf$Tumor_Seq_Allele2[i]!='T')){
    CtoT=CtoT+1
  }
}

TtoA=0
for (i in 1:length(BRCA.muse.maf$Tumor_Seq_Allele1)){
  if(!(BRCA.muse.maf$Tumor_Seq_Allele1[i]!='T') & !(BRCA.muse.maf$Tumor_Seq_Allele2[i]!='A')){
```

```
   TtoA=TtoA+1
 }
}

TtoG=0
for (i in 1:length(BRCA.muse.maf$Tumor_Seq_Allele1)){
 if(!(BRCA.muse.maf$Tumor_Seq_Allele1[i]!='T') & !(BRCA.muse.maf$Tumor_Seq_Allele2[i]!='G')){
   TtoG=TtoG+1
 }
}

TtoC=0
for (i in 1:length(BRCA.muse.maf$Tumor_Seq_Allele1)){
 if(!(BRCA.muse.maf$Tumor_Seq_Allele1[i]!='T') & !(BRCA.muse.maf$Tumor_Seq_Allele2[i]!='C')){
   TtoC=TtoC+1
 }
}
#Merge the values of all the 12 types of substitutions into one list
all_mut=c()
all_mut$AtoG=AtoG
all_mut$AtoC=AtoC
all_mut$AtoT=AtoT
all_mut$GtoA=GtoA
all_mut$GtoC=GtoC
all_mut$GtoT=GtoT
all_mut$CtoA=CtoA
all_mut$CtoG=CtoG
all_mut$CtoT=CtoT
all_mut$TtoA=TtoA
all_mut$TtoG=TtoG
all_mut$TtoC=TtoC

#Get the values of all the 12 types of substitutions into one vector
mut_val=c()
```

```
mut_type=c('A>G','A>C','A>T','G>A','G>C','G>T','C>A','C>G','C>T','T>A','T>G','T>C')  #vector representing the 12 mutation types

for(i in 1:length(all_mut)){

  mut_val=c(mut_val,all_mut[[i]])

}
```

After we got all the necessary data in vector form, we calculated the percentage frequency of all the 12 mutation types and plotted a bar plot of the resulting data.

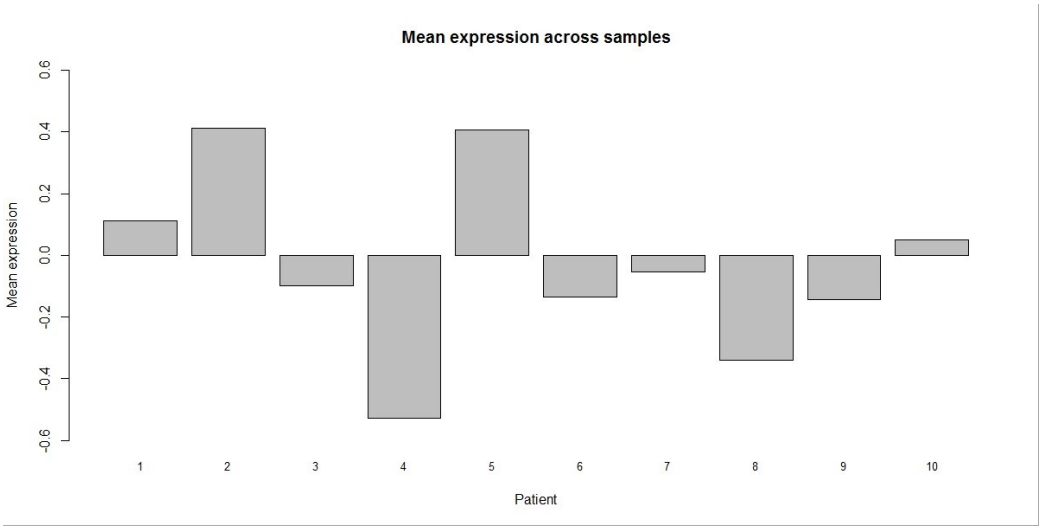
```
percentage_mut_val=mut_val/sum(mut_val)*100

mids=barplot(percentage_mut_val,col=c('red'),main="Point Mutation Spectra (BRCA)",

ylim=c(0,25),xlab="Mutations",ylab='Frequency [%]')

axis(1, at=mids, labels=mut_type,las=1,cex.axis=0.8)
```

Results

WGCNA

First, we showed the mean gene expression for each patient in a bar plot.

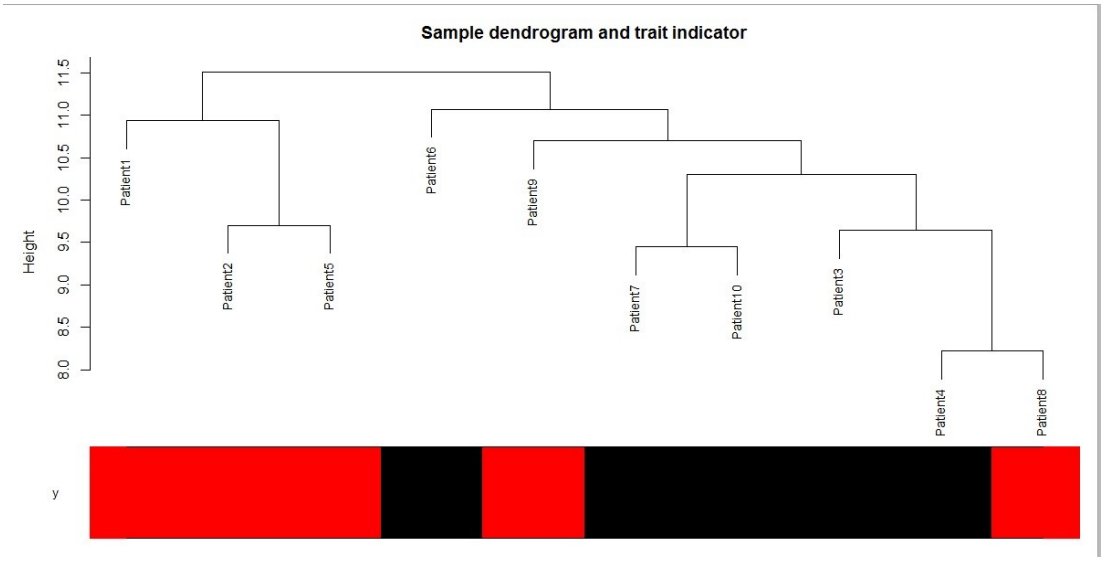
Figure 1:Bar Plot for Patient data (FPKM expression)



In this bar plot, we wanted to find if there was any sample with an outlying mean expression value. We see that no sample in the plot seem to have an outlying mean expression value.

After this, in order to find any outlier sample, we drew a Clustering dendogram of the samples.

Figure 2:Dendogram and trait Indicator



The dendogram showed samples divided into two traits (red and black). The dendogram shows no outliers in the samples.

Then, we related the genes to the trait data (obtained from the datTrait tutorial file) by marginal Pearson correlation. Then, we defined the Noise Gene Indicator values for the data, and consequently defined the Signal Gene Indicator values for the data by subtracting the Noise Gene Indicator values from 1. We saw that the mean value Noise Gene Indicator for the top 20 most significant genes was 0.85, which was very high, meaning that 85% of the top most significant 20 genes were noise, and only 15% were signals. This result was due to the small number genes in our data (53).

Figure 3:Noise (Fraction)

```
> mean(NoiseGeneIndicator[rank(p.Standard)<=20])
[1] 0.85
> mean(SignalGeneIndicator[rank(p.Standard)<=20])
[1] 0.15
>
```

We wanted to see that how many noise genes had a q-value less than or equal to 0.20. We found that none of the genes had a q-value less than or equal to 0.20, meaning that none of the genes had a q-value of greater than 0.2, meaning that all of them had more noise.

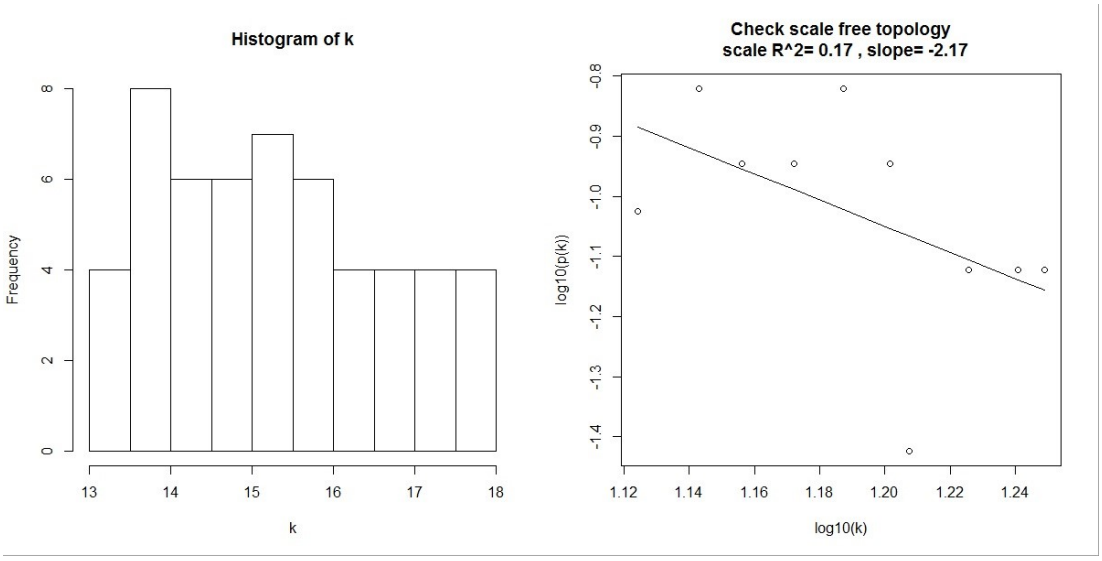
Figure 4:Genes at Noise q-value<=20

```
> table(q.Standard<.20)

FALSE
   53
>
> mean(NoiseGeneIndicator[q.Standard<=0.20])
[1] NaN
>
```

Finally, we went on to do the network construction for our data. According to our sample, we defined the adjacency matrix for our data and adjusted the soft power to accordingly, so that we could see a better representation in the future visualizations. We adjusted the power because after the correlation, the resulting values will be very small and the difference between them will be minute. So, if we enhance and magnify the values by adding a soft power of 10 or more (the result raised to the power of 10 or more), we can get a bigger difference between two values and hence can perform a better relation and comparison between them.

Next, we checked the Scale Free Topology of our data. We drew a histogram and log-log plot for the data of the histogram.

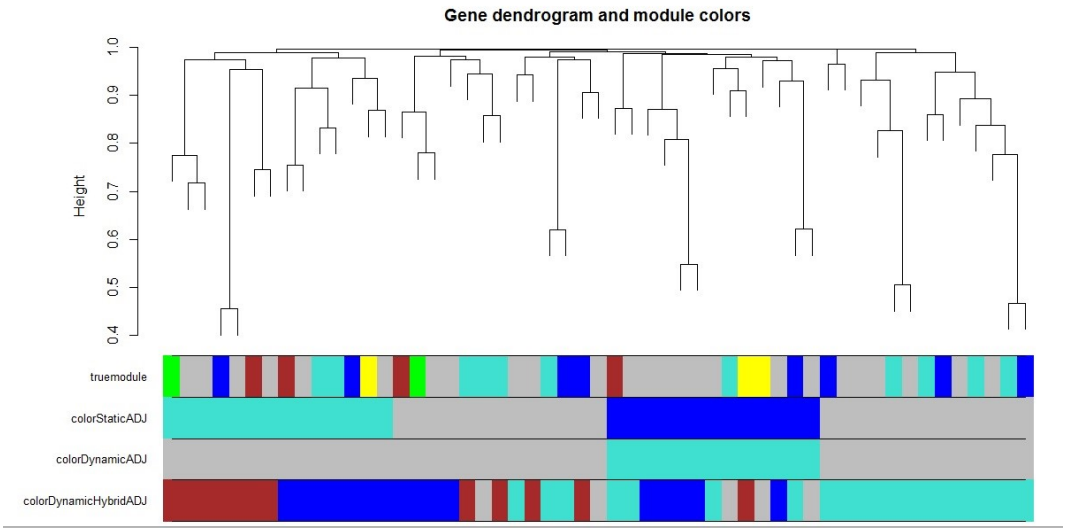
Figure 5:Histogram and Log-Log plot of Scale Free Topology



Our plot showed that even though there was a general approximation of the topology (straight line), there were no heavy outliers in our data.

After this, we compared different module detection methods for our data, in order to find the best module detection methods from the lot. Since our data was small (53 genes). We had to set the cut height of the tree and the minimum cluster size for all our modules.

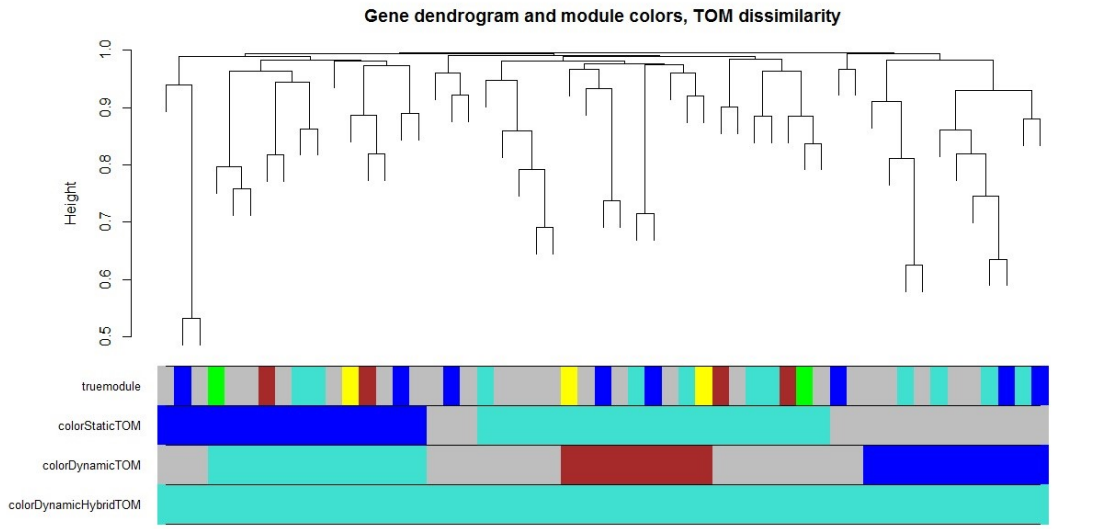
Figure 6:Gene dendogram with module colors



In this clustering, we used three different methods. The Figure shows that we could only detect 2 modules for the Static tree cut Method and only 1 module for the Dynamic tree cut method. We saw a lot of modules for the Dynamic Hybrid tree cut method.

Next, we wanted to see the module topology for the samples by using the topological overlap based dissimilarity. For this, we did hierarchical clustering of TOM-based dissimilarity for the genes. In simple words, the clusters were combined based on the measures of their dissimilarity. This method is the opposite of the one done in the previous Figure.

Figure 7:Gene dendogram with module colors (TOM dissimilarity)



In this clustering, we used three different methods. The Figure shows that we could only detect 2 modules for the Static dissTOM tree cut Method and only 3 modules for the Dynamic dissTOM tree cut method. We saw a big module for the Dynamic Hybrid dissTOM tree cut method.

Finally, we compared the different methods based on their “Rand index” and wanted to see the agreement between the different methods.

Figure 8:Cmparison of the different tree cut methods

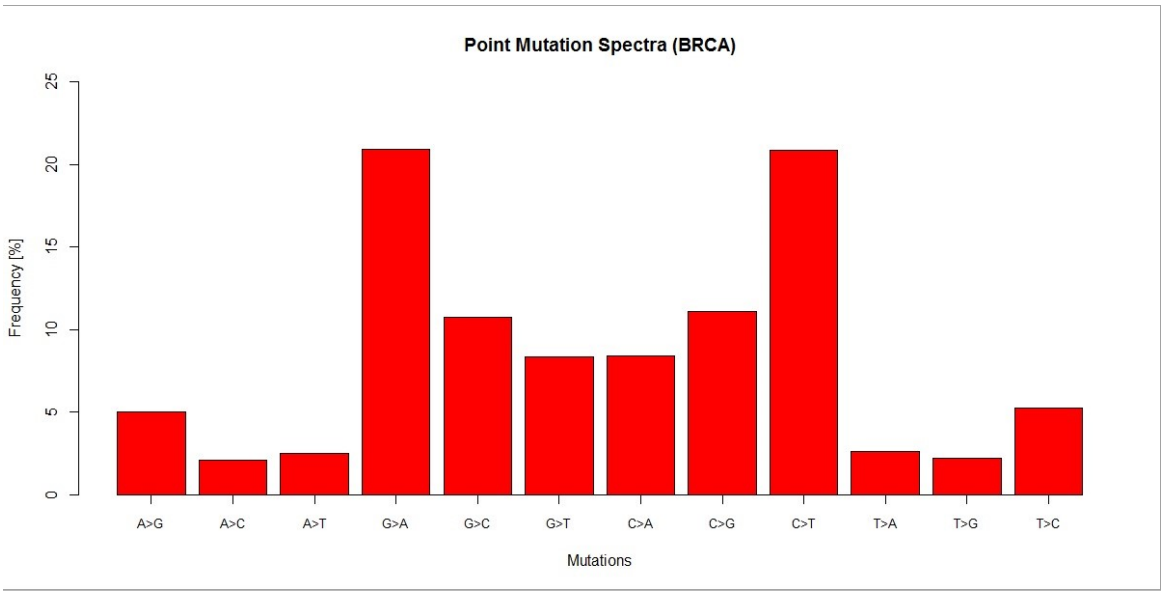
```
> randIndex(tabStaticADJ,adjust=F)
[1] 0.5595065
> randIndex(tabStaticTOM,adjust=F)
[1] 0.5624093
> randIndex(tabDynamicADJ,adjust=F)
[1] 0.4332366
> randIndex(tabDynamicTOM,adjust=F)
[1] 0.6023222
> randIndex(tabDynamicHybridADJ ,adjust=F)
[1] 0.5914369
> randIndex(tabDynamicHybridTOM ,adjust=F)
[1] 0.2851959
>
```

We saw that the dissTOM worked better for the first two cut tree methods. So, overall dissTOM was a better method for getting the cluster modules for our data.

Mutation Frequency and Spectra Analysis

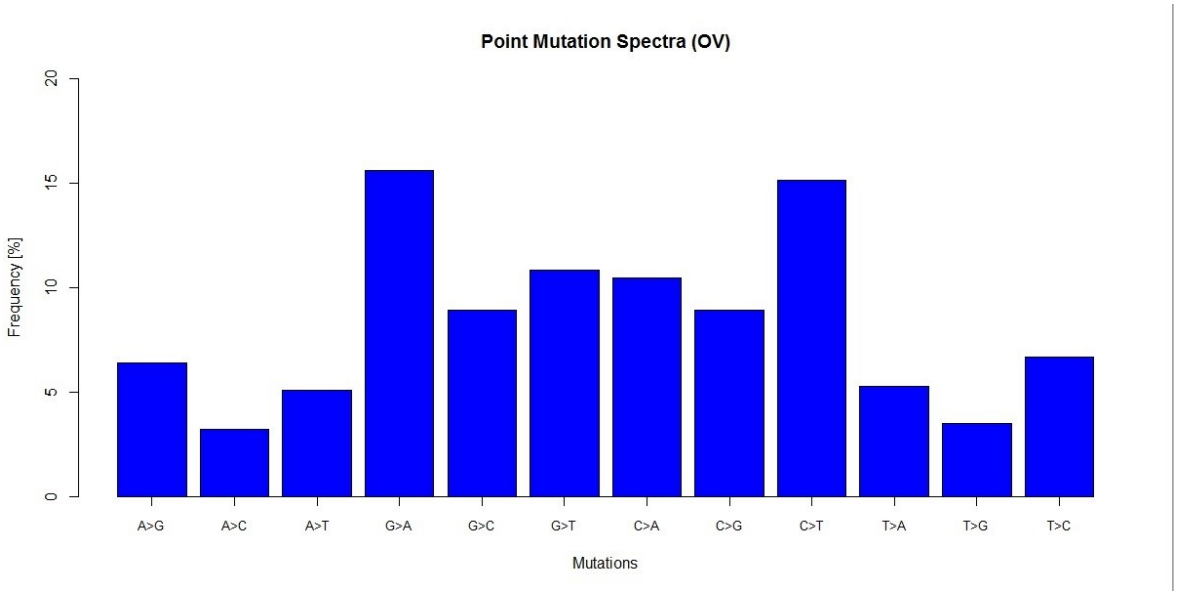
The mutation frequency data is shown in the supplemental .pdf files. The mutation spectra data for both BRCA and OV are shown below.

Figure 9:Mutation Spectra for BRCA



In this bar plot, we see that for BRCA, the G>A and C>T mutations have the highest mutation frequency percentage.

Figure 10:Mutation Spectra for OV



From this bar plot, we see that for OV, the G>A and C>T mutations have the highest mutation frequency percentage.

The data in both the bar plots suggests that the mutation spectra for both BRCA and OV is very similar and they show similar trends when it comes to the relative percentage frequency distribution for the mutation type, even though BRCA has relatively higher percentage of mutations for G>A, G>C, G>T and C>T but lower percentage for G>T and C>A.

Discussion

Our analysis was based on a small sample size that was just focusing on a small number of genes. The WGCNA analysis could have been better if we had a large data set of genes for a large number of patient samples. Also, it could be improved if we could include more types of cancers into the analysis. The coding done for this project could have been made more efficient with the use of in-built functions or custom user-defined functions. The logic could have been improved by tackling problems with a more efficient and less time-consuming way.