University of Tampere

BioMedTech

High throughput data analysis

By:Nigatu Adossa

Data: June, 2014

Contents

1. Gene expression analysis of prostate cancer 3
2. RNA-seq data analysis 16
3. CHIP-Seq data analysis 18
4. Studying the mechanisms of androgen insensitivity 20
5. Appendixes 21
6. Gene expression analysis of prostate cancer

**Task 1. Loading the data matrix to the R environment.**

sampleData<-read.table("gene\_expr\_microarrays/sample\_data.txt", row.names=1,sep="\t")

sampleHeader<-read.table("gene\_expr\_microarrays/sample\_header.txt", skip=2, header=FALSE,sep="\t")

sampleId<-unlist(sampleHeader[1,2:83])

colnames(sampleData)<-sampleIdcolnames(sampleData)<-sampleId

Now the data matrix is ready for further analysis with columns of the sample names in the “sample\_header.txt”file.

**Task 2. Summarization**

The dimension of the data matrix before pre processing is:

> dim(sampleData)

[1] 54675 82

And the row identifiers are a probe ids:

head(rownames(sampleData))

[1] "1007\_s\_at" "1053\_at" "117\_at" "121\_at" "1255\_g\_at" "1294\_at"

We want to change these probe identifiers by the gene symbol each probe identifier corresponds with. As we are trying to change the probe identifier with the gene names, we faced a problem of finding several probes mapping to the same gene and one probe identifier mapping to several genes. Therefore, we have to first consider multiple probe ids mapping to the same gene and find the average expression for all its samples and replace the expression values of each probe with this newly calculated value. In order to perform this task, we need to download annotation database of the particular data matrix.

source("http://bioconductor.org/biocLite.R")

biocLite("hgu133plus2.db")

library("hgu133plus2.db")

Extracting all gene names annotated for all the probe identifiers in our raw data matrix from the database, we can precede our data pre processing steps.

geneName<-select(hgu133plus2.db, probIds, "ALIAS" , "PROBEID")

Now let us filter the probeIds that does not have duplicated occurrence in our lists of gene name.

notDuplicatedGenebyprobyid<-geneName[!duplicated(geneName[,1]),]

> dim(notDuplicatedGenebyprobyid)

[1] 54675 2

Checking the filtered genes:

> head(notDuplicatedGenebyprobyid)

PROBEID ALIAS

1 1007\_s\_at MIR4640

15 1053\_at RFC2

17 117\_at HSPA6

18 121\_at PAX8

19 1255\_g\_at C6orf131

28 1294\_at D8

Now we got unique probe ids mapping to unique gene names. The next step is to find the expression values for each of these unique genes. To get the expression values for each of the samples of filtered genes, we need to do summarization process in which we take the mean value of expression for different probes mapped to the same gene and assign the mean expression value to the not duplicated genes and probe identifier. The following code does the summarization process and assigns the summarized expression values to the newly created data matrix.

notDuplicatedByGeneNameProbId<-notDuplicatedByProbId[!duplicated(notDuplicatedByProbId[,2]),]

> dim(notDuplicatedByGeneNameProbId)

[1] 20126 2

newDataMatrix<-matrix(nrow=dim(notDuplicatedByGeneNameProbId)[1],ncol=length(colnames(sampleData)))

for(i in 1:dim(notDuplicatedByGeneNameProbId)[1]){

#search for all duplicated genes associated with the nonduplicated gene

dupProbId<-notDuplicatedByProbId[grep(notDuplicatedByGeneNameProbId[i,2],notDuplicatedByProbId[,2]),1]

#replace the expression value by the avarage expression values of those duplicated genes for each sample

newDataMatrix[i,]<-apply(sampleData[dupProbId,],2,mean)

}

Now we completed our summarization step and created a new data matrix called summerizedSampleData

>summerizedSampleData<- newDataMatrix

Finally lets give the column names representing each of the samples and row names representing each of the not duplicated gene names.

>rownames(summerizedSampleData)<- notDuplicatedGenebyprobyid [,2]

>colnames(summerizedSampleData)<-sampleId

Now we can check the expression values of gene using its names and corresponding samples:

> summerizedSampleData["MIR4640","GSM38079"]

[1] 3186.727

**Task 3. Preprocessing**

The next step is to log transform the values in the data matrix. Just to compare how the transformation will affect the data matrix lets first look the boxplot and distribution of the summarized sample data.

> plot(summerizedSampleData)

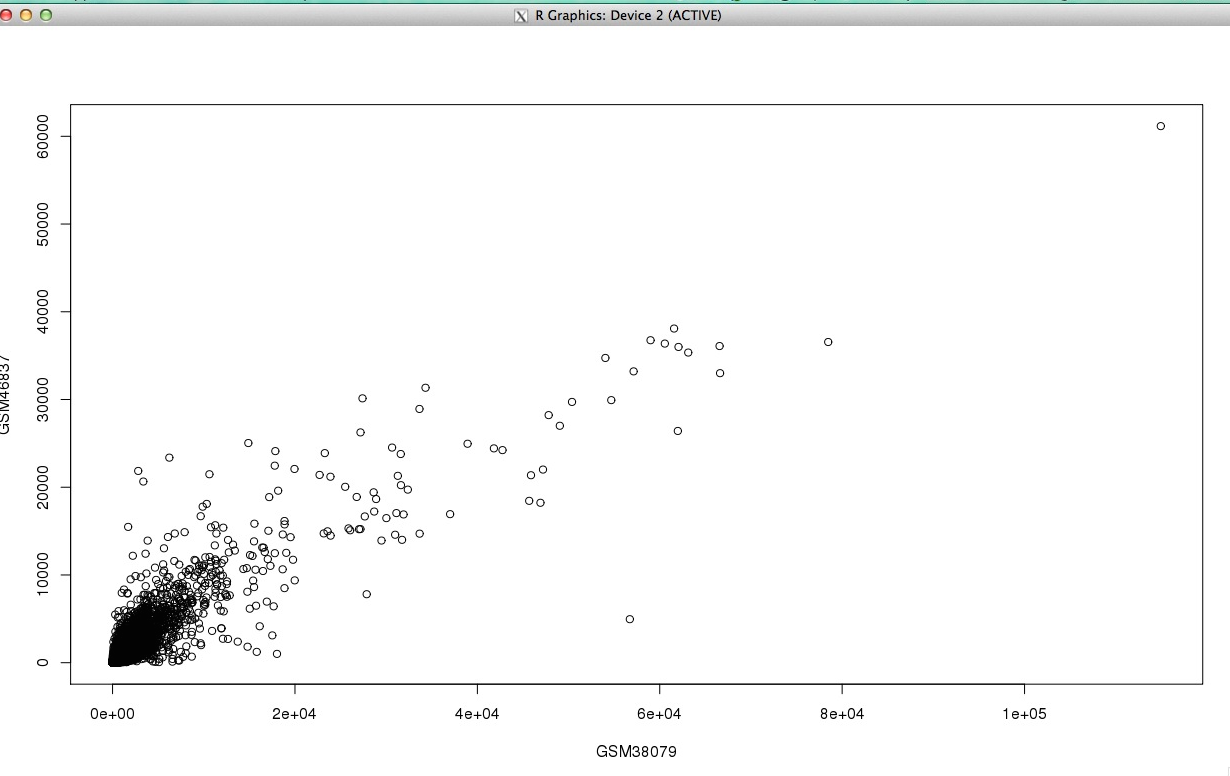


Figure 1. A plot of the summarized sample data

>boxplot(summerizedSampleData)

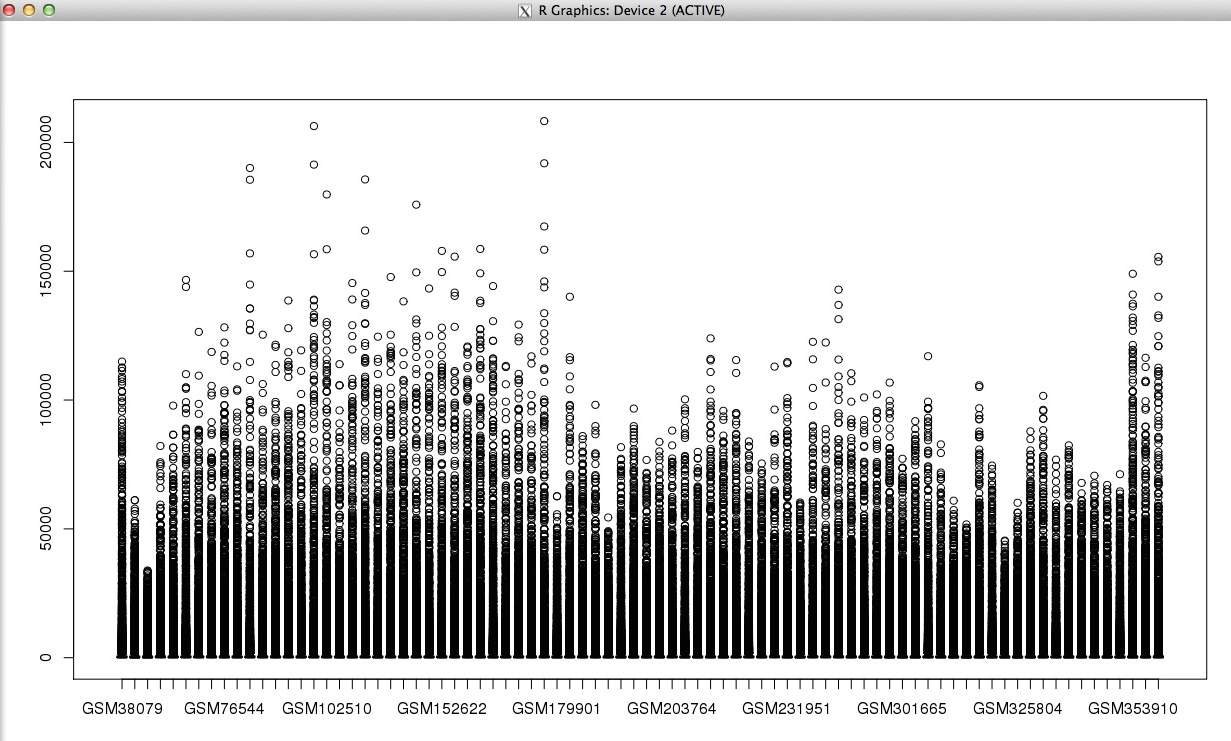


Figure 2. Boxplot of the summarized data matrix

> hist(summerizedSampleData)

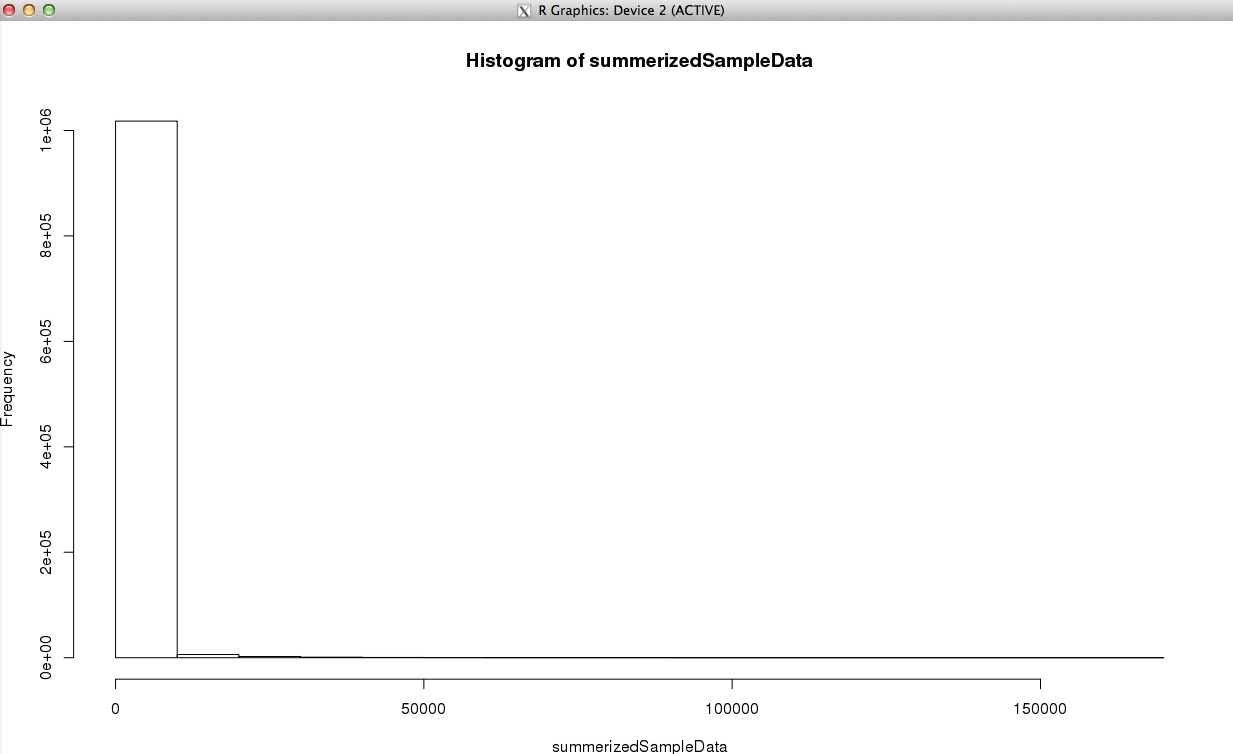


Figure 3. Histogram of the summarized data matrix

As we can see from the figure 3, most of the unnormalized values are concentrated around zero below zero.

Log transforming the summarized data matrix we get the following plots:

> plot(logSummerizedSampleData)

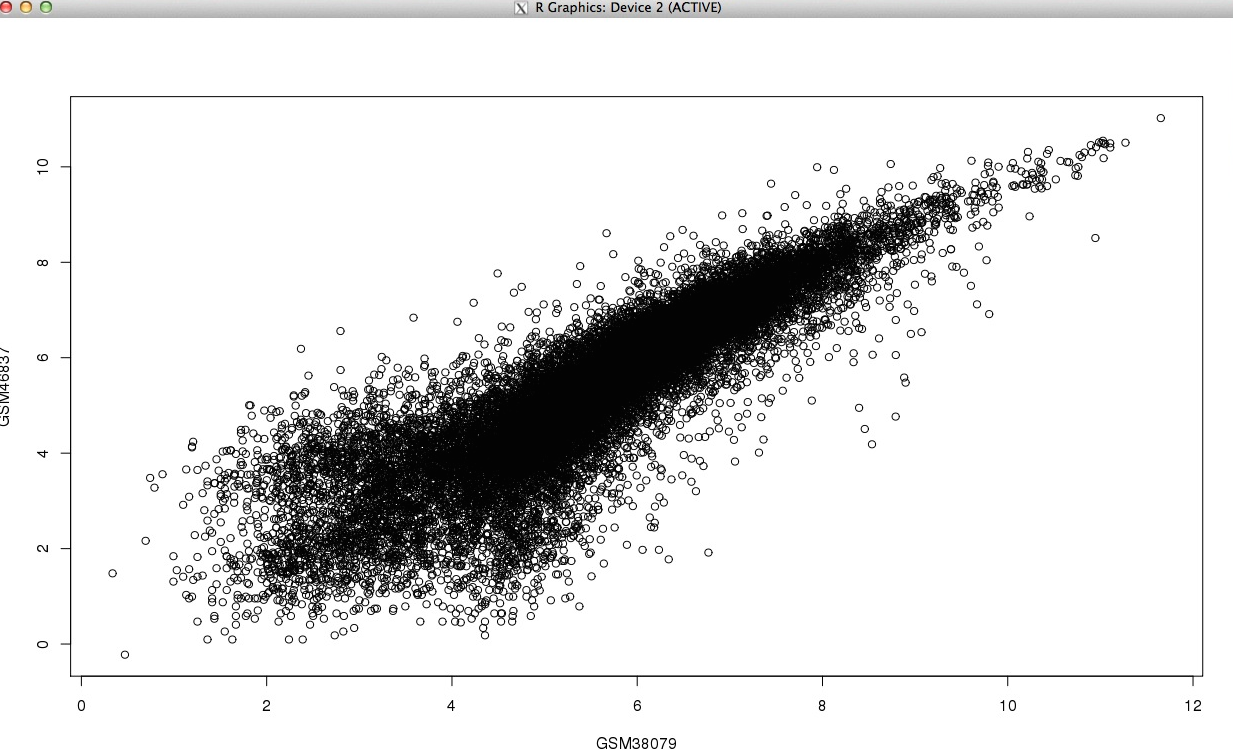


Figure 4. A plot of log transformed summarized data

>logSummerizedSampleData<-log(summerizedSampleData,2)

>boxplot(logSummerizedSampleData)# after log transformation

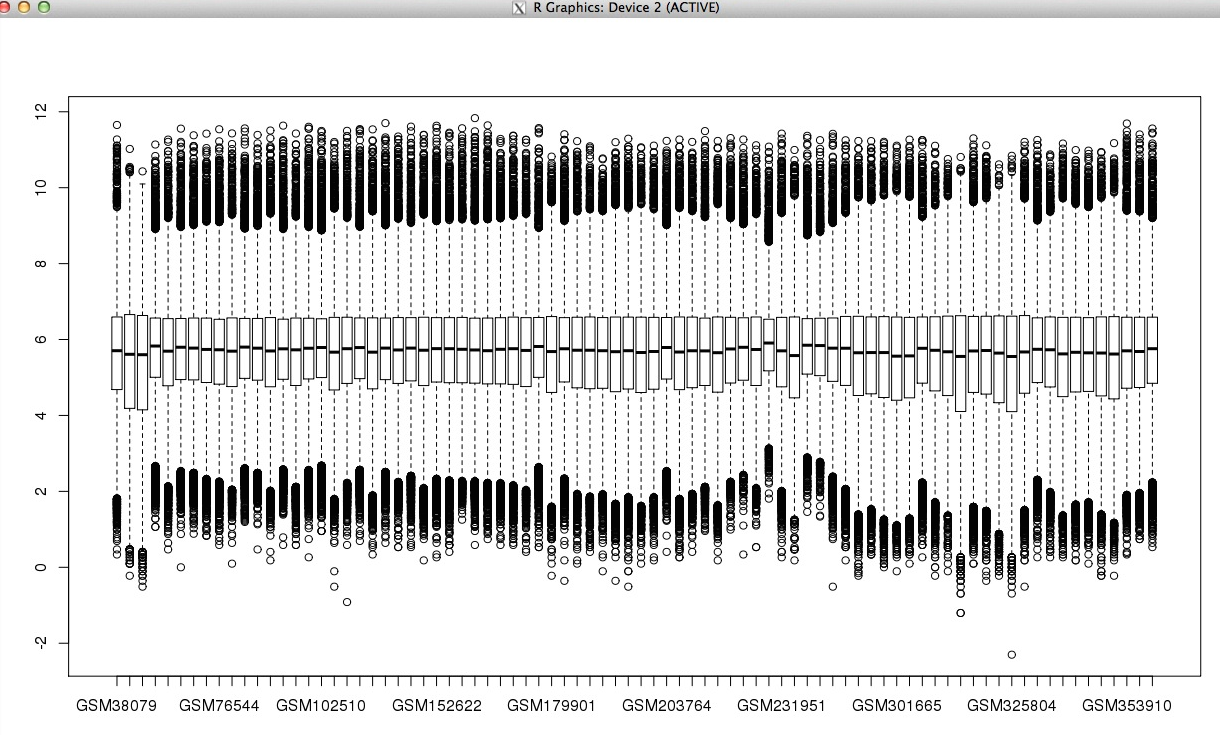


Figure 5. Boxplot of log transformed data matrix

As we can see from figure 5, there is a slight fluctuation around the mean vales and upper and lower quartiles. Therefore, normalizing the log transformed matrix will improves the data quality. The quartile normalization is used in the normalization process using the “preprocessCore” package in R.

>library(preprocessCore)

>normalizedLogSummerizedSampleData<-normalize.quantiles(as.matrix(logSummerizedSampleData),copy=TRUE)

>rownames(normalizedLogSummerizedSampleData)<-rownames(logSummerizedSampleData)

>colnames(normalizedLogSummerizedSampleData)<-colnames(logSummerizedSampleData)

>boxplot(normalizedLogSummerizedSampleData)

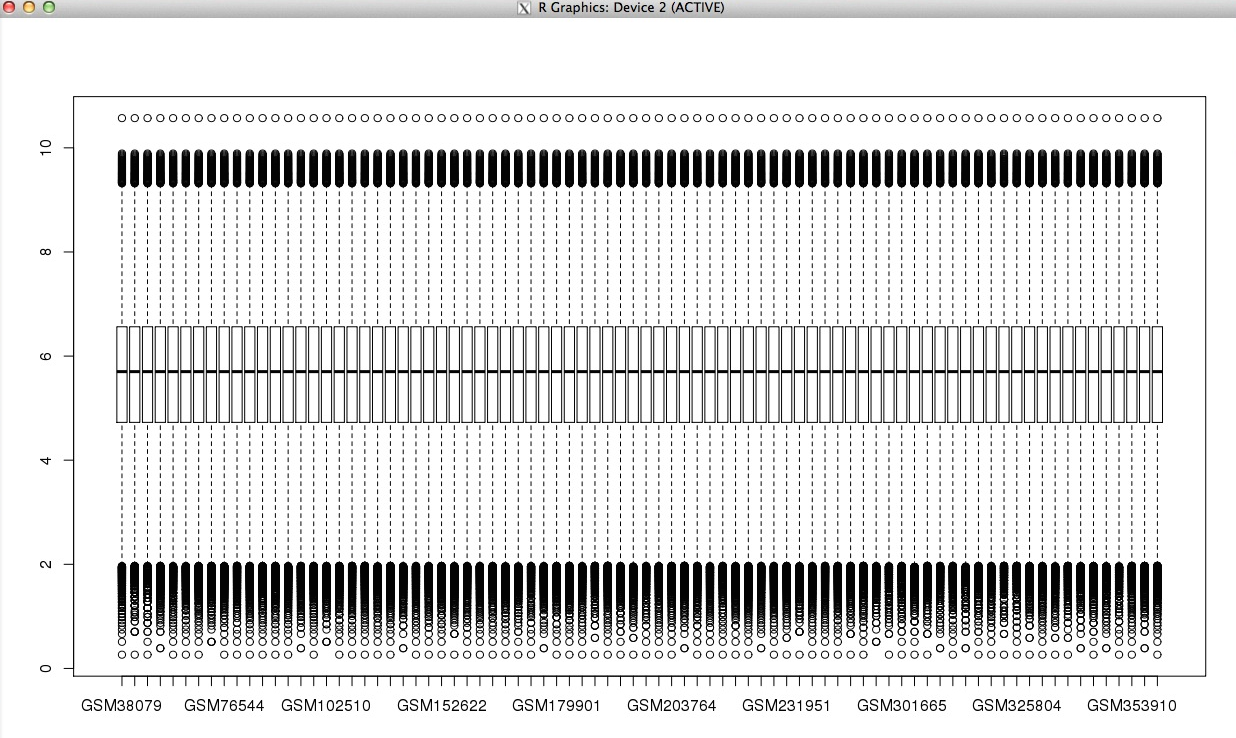


Figure 6. Normalized data matrix

We can also see the distribution of the normalized data is more of Gaussian using the following histogram.

>boxplot(normalizedLogSummerizedSampleData)

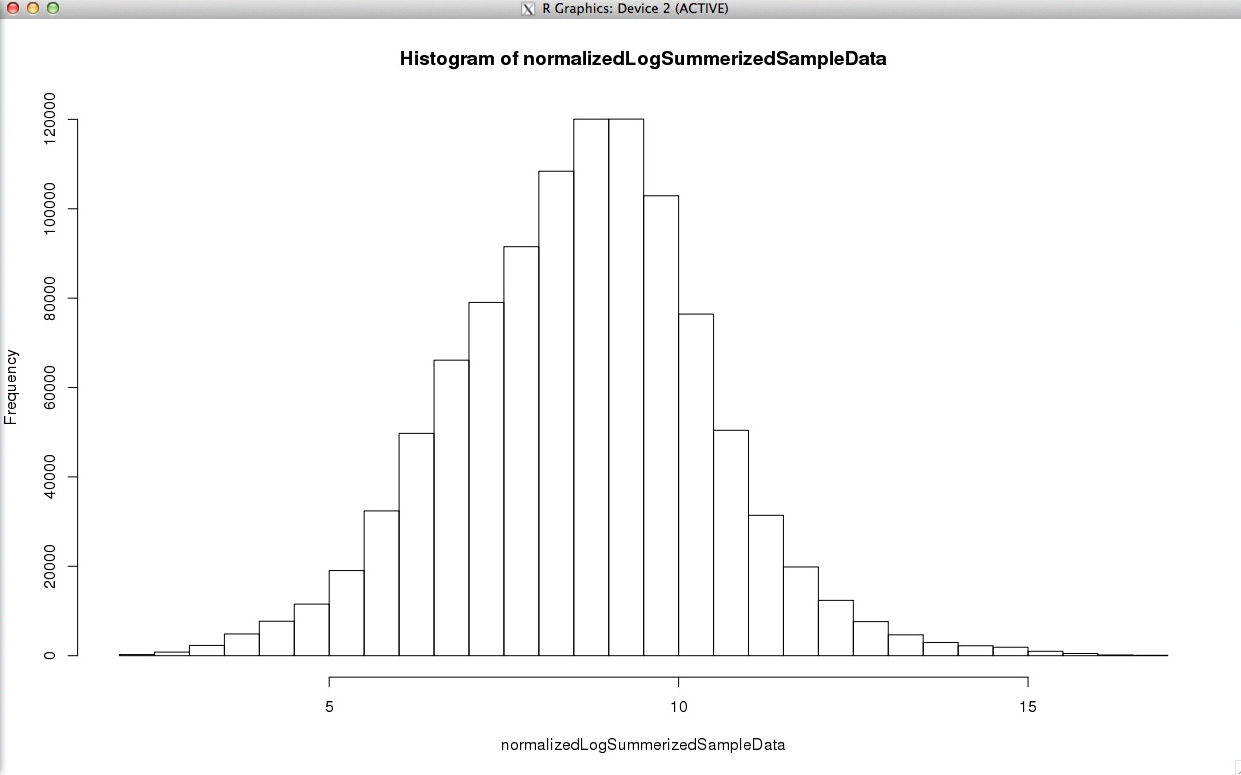


Figure 7. Histogram of normalized data matrix

With this, we have well enough normalized data matrix and completed the pre-processing and we are in a good condition to proceed to the next data analysis steps.

**Task 4. Sample class labels**

As we summarize and pre processed the original data matrix, it is obvious that there will be a probability not to get ETV1 & ETV4 from the preprocessed matrix gene lists. These are because we summarized the probe mapping to more than one gene to one probe. Now the trick is to find the gene which representing the ETV1 & ETV 4 in our processed and normalized matrix and check the expression level of that gene.

Since one probe id is mapped for several gene names in in the original data matrix, if we find the probe id representing ETV1 & ETV2, then based on that we can search similar probe ides in our list of unique probe id and gene names. Then the gene name that we get from our search is going to be the accession name for ETV1 and ETV4.

>ETV1ProbId<- intersect(geneName[which(geneName$ALIAS=="ETV1"),1],notDuplicatedByGeneNameProbId[,1])

> ETV1ProbId

[1] "206501\_x\_at"

>ETV1GeneinSummerizedGeneSet<-notDuplicatedByGeneNameProbId[notDuplicatedByGeneNameProbId$PROBEID==ETV1ProbId,2]

**> ETV1GeneinSummerizedGeneSet**

**[1] "ER81"**

This means both ER81 from the processed matrix & ETV1 from not processed matrix have the same probe id. As the new preprocessed and normalized data matrix uses gene name as an accession (row name), I can use ER81 in sated of ETV1. For example, if I want the expression value of ETV1 of sample **“GSM53061”:**

**> normalizedLogSummerizedSampleData["ER81", "GSM53061"]**

**[1] 6.866241**

This means I got the expression of ETV1 using gene with same probe id as ETV1. In the same way we can get the expression values of the other genes which are not in our list of preprocessed data matrix.

ETV4ProbId<- intersect(geneName[which(geneName$ALIAS=="ETV4"),1],notDuplicatedByGeneNameProbId[,1])

ETV4GeneinSummerizedGeneSet<-notDuplicatedByGeneNameProbId[notDuplicatedByGeneNameProbId$PROBEID==ETV4ProbId,2]

EST\_Rearrangenes<-c(ETV1GeneinSummerizedGeneSet,ETV4GeneinSummerizedGeneSet)

> EST\_Rearrangene

[1] "ER81" "E1A-F"

Finally we can plot to see the expression variation of each gene:

> plot(normalizedLogSummerizedSampleData[ETV1GeneinSummerizedGeneSet,])

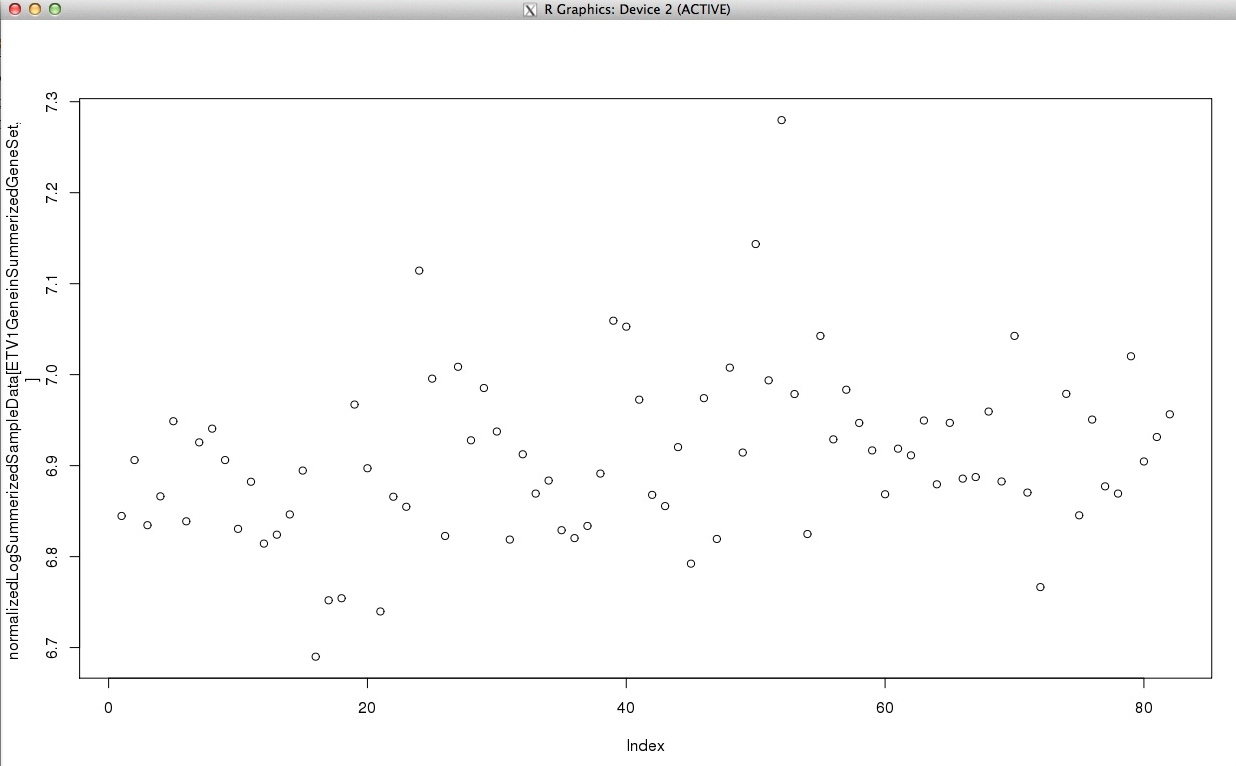


Figure 8. Plot of the ETV1 expression values.

As it is shown in the figure 8, the average expression level seems 6.9. Therefore samples that show expression of greater than 7.1 are considered to be an overexpressed samples of ETV1 gene.

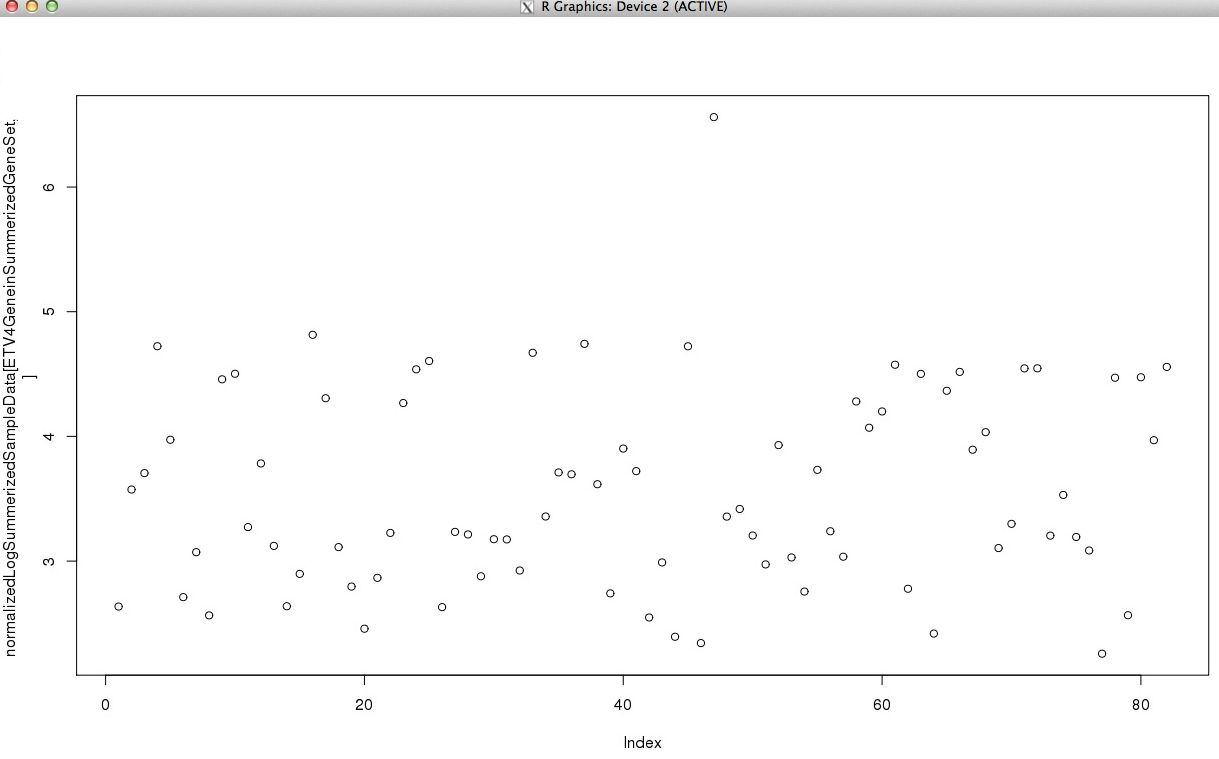
>highlyExpressdETV1<-names(which(normalizedLogSummerizedSampleData[EST\_RearrangeProbIds[1],]>7.1))

**> highlyExpressdETV1**

**[1] "GSM152611" "GSM231888" "GSM231944"**

In the same way we can plot the expression values of the ETV4:

**>plot(normalizedLogSummerizedSampleData[ETV4GeneinSummerizedGeneSet,])**

****

**Figure 9. Expression values of ETV4**

Since the average value is around 4.5, samples that have an expressions value of greater than 6 is considered overexpressed.

**> highlyExpressdETV4<-names(which(normalizedLogSummerizedSampleData[EST\_RearrangeProbIds[2],]>6))**

**> highlyExpressdETV4**

**[1] "GSM231872"**

Now we got the sample groups that show over expression of EST gene families.

highlyExpressedEST\_RearrangeSamples<-c(highlyExpressdETV1,highlyExpressdETV4)

**> highlyExpressedEST\_RearrangeSamples**

**[1] "GSM152611" "GSM231888" "GSM231944" "GSM231872"**

In the same way we also get samples with over expression of SPINK1, plotting the values to select the threshold value:

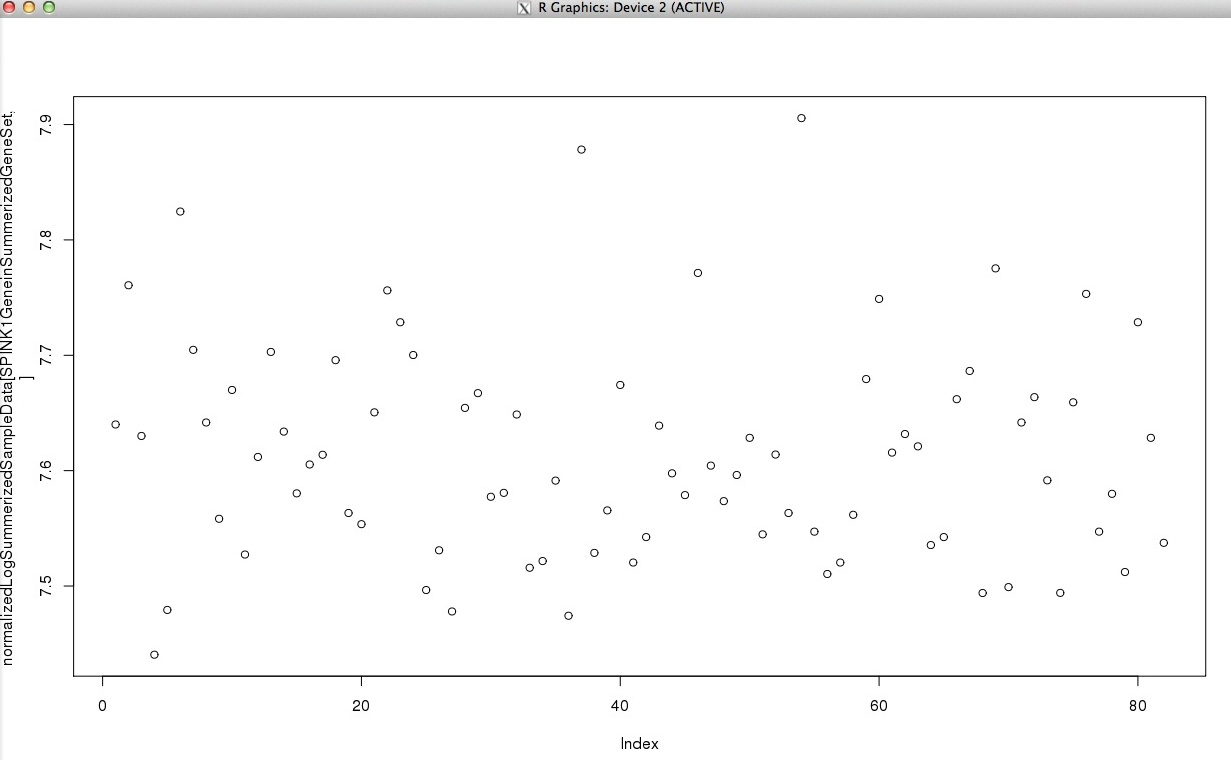


Figure 10. Plot of SPINK1 gene

Then we select 7.8 as overexpression threshold value:

>SPINK1ProbId<- intersect(geneName[which(geneName$ALIAS=="SPINK1"),1],notDuplicatedByGeneNameProbId[,1])

>SPINK1GeneinSummerizedGeneSet<-notDuplicatedByGeneNameProbId[notDuplicatedByGeneNameProbId$PROBEID==SPINK1ProbId,2]

>highlyExpressedSPINK1Sample<-names(which(normalizedLogSummerizedSampleData[SPINK1GeneinSummerizedGeneSet,]>7.8))

>highlyExpressedSPINK1Sample<-names(which(normalizedLogSummerizedSampleData[SPINK1ProbId,]>7))

> highlyExpressedSPINK1Sample

[1] "GSM53152" "GSM179954" "GSM231957"

And the remaining sample groups are the ones those do not belong to either of the two samples.

>allSamples<-colnames(normalizedLogSummerizedSampleData)

>knownSamples<-c(highlyExpressedSPINK1Sample,highlyExpressedEST\_RearrangeSamples)

>unknownSamples<-setdiff(allSamples,knownSamples)

> unknownSamples

[1] "GSM38079" "GSM46837" "GSM46866" "GSM53061" "GSM53114" "GSM53162"

[7] "GSM76516" "GSM76544" "GSM76553" "GSM76640" "GSM76648" "GSM88977"

[13] "GSM89017" "GSM102435" "GSM102498" "GSM102510" "GSM117726" "GSM117727"

[19] "GSM117741" "GSM137971" "GSM138038" "GSM152575" "GSM152617" "GSM152622"

[25] "GSM152631" "GSM152772" "GSM152778" "GSM152783" "GSM179790" "GSM179792"

[31] "GSM179843" "GSM179849" "GSM179901" "GSM179903" "GSM203677" "GSM203707"

[37] "GSM203711" "GSM203715" "GSM203722" "GSM203740" "GSM203764" "GSM203778"

[43] "GSM203786" "GSM231876" "GSM231881" "GSM231894" "GSM231951" "GSM231978"

[49] "GSM231979" "GSM231990" "GSM277677" "GSM277683" "GSM277694" "GSM301659"

[55] "GSM301665" "GSM301666" "GSM301670" "GSM301674" "GSM301679" "GSM301701"

[61] "GSM301709" "GSM325799" "GSM325802" "GSM325804" "GSM325810" "GSM353882"

[67] "GSM353884" "GSM353891" "GSM353892" "GSM353893" "GSM353894" "GSM353899"

[73] "GSM353910" "GSM353917" "GSM353940"

**Task 5 Deferentially expressed genes**

To calculate the differently expressed genes, we use R package called lima. First we load the package to the R environment:

biocLite("limma")

library("limma")

Lima uses a model based differential expression approach, were we fir our model matrix to the model. To do so we first create a class level or sub groups of the samples of the genes. To do so, we first create a vector of 82 elements with value 1 and we replace our sub groups, EST and SPINL1 sub groups with 2 and 3 respectively. As a result we get 1 representing the unknown sumple group, 2, from EST sub group and 3 for SPINL1 sub groups.

samples<-c(rep(1,82))

EST\_RearrangeSamples<-match(highlyExpressedEST\_RearrangeSamples,colnames(normalizedLogSummerizedSampleData))

SPINK1Sample<-match(highlyExpressedSPINK1Sample,colnames(normalizedLogSummerizedSampleData))

sample[EST\_RearrangeSamples]<-2

samples[SPINK1Sample]<-3

**> samples**

**[1] 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 3 1**

**[39] 1 1 1 1 1 1 1 1 2 1 1 2 1 2 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1**

**[77] 1 1 1 1 1 1**

Once we created factored samples, we can create a model matrix to fit in the lima package’s model.

>design <- model.matrix(~0+factor(samples))

Giving the descriptive name to the three sub group:

>colnames(design)<-c("unknownSample","EST\_Samples","SPINK1Sample")

Then we fit our model with the our processed data matrix together with our class subgroups

fit <- lmFit(normalizedLogSummerizedSampleData, design)

Since we want to compare each of the sub group’s deferential expression with each other, we can create a contrast matrix so that the lima can do it automatically.

Using conf parameter we can specify the subclass that we want to check for differential expression, 1 is for unknown subgroup and EST sub group, 2 for SPINK1 and EST and 3 is for SPINK1 and unknown subclasses.

**top\_DE\_gene\_unknown\_EST<-topTable(fit2, coef=1, adjust="BH")**

Lists of the differentially expressed genes between the unknown subclass and EST\_Samples subclases

**> top\_DE\_gene\_unknown\_EST**

**logFC AveExpr t P.Value adj.P.Val B**

**226202\_at 1.730264 6.036999 6.241055 1.849873e-08 0.0005081433 7.521244**

**220991\_s\_at 2.844325 4.213390 6.239940 1.858811e-08 0.0005081433 7.517541**

**209940\_at 2.094332 5.561251 5.970020 5.922189e-08 0.0010792991 6.625709**

**1553918\_at 2.435410 4.543259 5.861144 9.404573e-08 0.0012854641 6.268780**

**227032\_at 1.380115 5.671266 5.714795 1.742705e-07 0.0018297413 5.791936**

**218183\_at 1.402816 5.727074 5.645982 2.324389e-07 0.0018297413 5.568982**

**223960\_s\_at 1.673702 5.684919 5.644108 2.342647e-07 0.0018297413 5.562923**

**234072\_at 1.665770 5.513432 5.321931 8.855320e-07 0.0060519469 4.531353**

**240198\_at 1.563650 5.350824 5.281860 1.042426e-06 0.0063326209 4.404587**

**219259\_at 1.474218 5.562828 5.139126 1.855582e-06 0.0101452102 3.956110**

From the above lists of significantly expressed genes, if we set the significant threshold to be 0.05, from the adjusted p-value list**,** we got all the top 10 differentially expressed genes between the unknown sample groups and the ESt gene family groups are statistically significantly differently expressed.

**>top\_DE\_gene\_SPINK1\_EST<-topTable(fit2, coef=2, adjust="BH")**

List of top 10 differentially expressed genes between SPLINK1 subclass and EST\_Samples subclass.

**> top\_DE\_gene\_SPINK1\_EST**

**logFC AveExpr t P.Value adj.P.Val B**

**227159\_at -0.9411074 8.206142 -4.900756 4.783437e-06 0.08083454 2.344076**

**215556\_at -2.1920842 5.462408 -4.898210 4.831512e-06 0.08083454 2.337367**

**242888\_at 2.2046845 5.869110 4.786149 7.484595e-06 0.08083454 2.043482**

**1553918\_at 2.7651298 4.543259 4.765500 8.108879e-06 0.08083454 1.989641**

**201551\_s\_at -0.8827150 8.842866 -4.751902 8.547364e-06 0.08083454 1.954240**

**201553\_s\_at -0.9570686 9.722639 -4.714634 9.870744e-06 0.08083454 1.857440**

**209940\_at 2.3035695 5.561251 4.702343 1.034938e-05 0.08083454 1.825589**

**240198\_at 1.9058225 5.350824 4.610122 1.473504e-05 0.10070293 1.587791**

**228931\_at 2.2337099 5.659949 4.558147 1.795320e-05 0.10198757 1.454725**

**226202\_at 1.7607428 6.036999 4.548041 1.865376e-05 0.10198757 1.428931**

From the above top 10 genes lists of the SPLINK1 sample groups and EST sample groups, based on the adjusted p-values, there are no gene which is statistically significantly expressed. This is expected result since both of the sample groups are characterized by over expression of their corresponding genes, they both have a high level of expression therefore they can not be statistically significantly differently expressed.

**>top\_DE\_gene\_SPINK1\_unknown<-topTable(fit2, coef=3, adjust="BH")**

The top 10 differentially expressed genes between SPINK1 subclass and the unkown subclass.

**> top\_DE\_gene\_SPINK1\_unknown**

**logFC AveExpr t P.Value adj.P.Val B**

**215556\_at -1.9347522 5.462408 -6.037023 4.448800e-08 0.002066137 7.677424**

**1556633\_at -1.6668877 5.206357 -5.912689 7.558026e-08 0.002066137 7.227556**

**228338\_at -1.3357108 5.465368 -5.816064 1.137936e-07 0.002073850 6.879993**

**205758\_at 0.6083410 6.958469 5.627948 2.506071e-07 0.002656527 6.208928**

**1555580\_at -2.5822998 4.385606 -5.584495 3.003188e-07 0.002656527 6.055041**

**241471\_at -2.4584024 4.668799 -5.535481 3.680788e-07 0.002656527 5.881996**

**214214\_s\_at 0.8920238 7.800376 5.523247 3.872111e-07 0.002656527 5.838892**

**208910\_s\_at 0.8924946 7.800958 5.522315 3.887080e-07 0.002656527 5.835610**

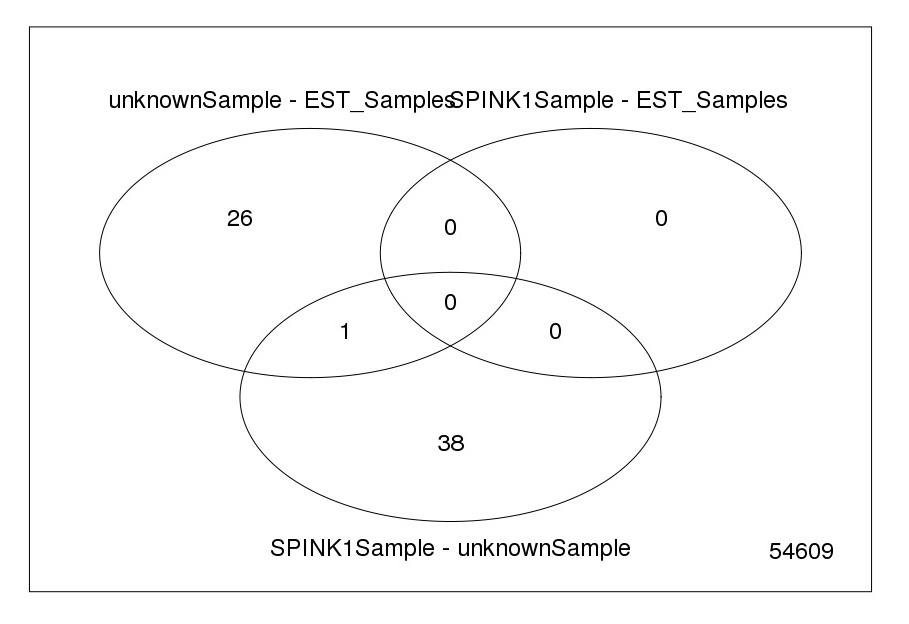
**1569736\_at -1.6736662 4.999747 -5.399469 6.448929e-07 0.003917652 5.404890**

**203157\_s\_at 1.1549174 5.746422 5.269659 1.095390e-06 0.005851462 4.954029**

As we see it with the differentially expression of EST gene family sample group and the unknown sample groups, the SPINKI and unknown gene sample group has all of the top 10 differentially expressed genes are differently expressed.

Lima also has a function called “decideTests” that decides differentially expressions numbers between all sample subclasses and shows it in a vain diagram.

**>results <- decideTests(fit2)**

**>vendiagram(results)**

**Figure 11. Vain diagram showing the significantly differently expressed number of genes.**

Based on the result of the comparison between the number of significantly differentially expressed genes among each of the three sub groups, we can say that our hypothesis is correct as there is relatively high number of significantly differently expressed genes between the unknown subgroup, which is the subclass of an average or low expression levels, and SPINK1 subclasses which has a high expression levels of SPINK1 gene. But we cannot warranty it because we don’t know weather it is due to the biological variations between the samples or it is due to differential expression. This can be confirmed after performing an enrichment analysis.

When we see the number of genes of the significantly differentially expressed genes between ESTSamples and unknown samples, then we also get 24 significantly expressed genes. Again to check this numbers are due to real differentially expression we have to do enrichment analysis.

The third comparison between the two highly expressed gene of EST gene family and SPINK1 gene respectively, we cannot get any number of statistically differential expressed genes. This confirms that the two subgroups are similar in their expression.

Putting all of the three comparisons together, and our hypothesis that EST and SPINK1 might be the cause for the prostate cancer.

**Task 6.** ETS-positive group

Loading the cdna\_microarry data’s to the R environment:

#cdna\_microarray

CRPC\_278<-read.csv("cdna\_microarrays/CRPC\_278.csv", skip=1, sep="\t")

CRPC\_543<-read.csv("cdna\_microarrays/CRPC\_543.csv", skip=1, sep="\t")

PCaN\_5934<-read.csv("cdna\_microarrays/PCaN\_5934.csv", skip=1, sep="\t")

PCaN\_6102<-read.csv("cdna\_microarrays/PCaN\_6102.csv", skip=1, sep="\t")

PCaP\_17163<-read.csv("cdna\_microarrays/PCaP\_17163.csv", skip=1, sep="\t")

PCaP\_470<-read.csv("cdna\_microarrays/PCaP\_470.csv", skip=1, sep="\t")

Combining all of the six samples data in to one data matrix, we can make a data frame of all gene symbol and expression values for each patient:

petients<-data.frame(gene=CRPC\_278[,1],

petients1=CRPC\_278[,2],

petients2=CRPC\_543[,2],

petients3=PCaN\_5934[,2],

petients4=PCaN\_6102[,2],

petients5=PCaP\_17163[,2],

petients6=PCaP\_470[,2])

> head(petients)

gene petients1 petients2 petients3 petients4 petients5 petients6

1 A1CF 6.39 9.17 4.59 1.42 0.85 -0.15

2 A2BP1 -0.74 4.32 8.83 8.68 6.98 4.98

3 A2LD1 7.33 4.38 6.96 6.34 7.78 7.78

4 A2M 14.01 17.76 17.35 18.29 17.27 15.67

5 A2ML1 2.04 8.36 7.66 6.42 6.64 7.47

6 A4GALT 7.79 10.17 10.74 10.59 10.90 10.05

To visualize all the samples expression as a histogram in a single graph, we can use the “ggplot” and “reshape” package in R.

>install.packages("ggplot2")

>install.packages("reshape")

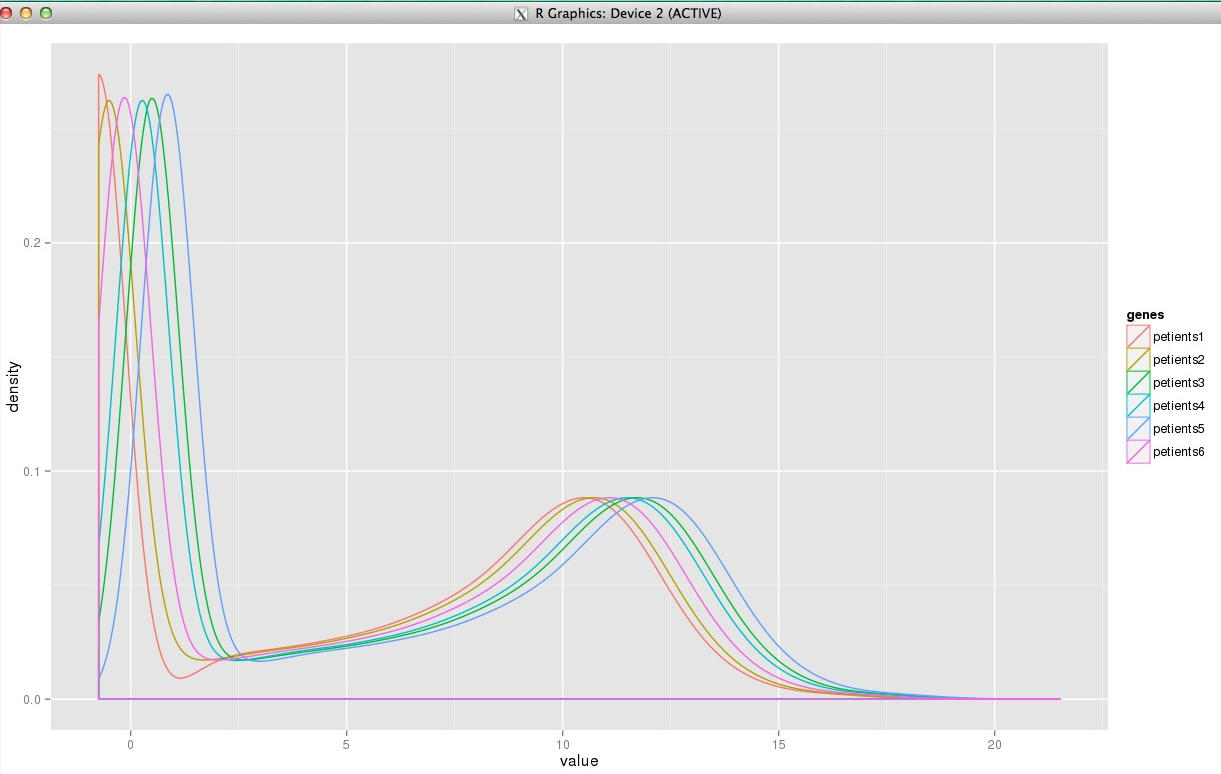
>library(reshape)

**>library(ggplot2)**

petients <- melt(petients, id = 'gene', variable\_name = 'genes')

histComparision<-ggplot(petients, aes(value,colour=genes)) + geom\_density()

>histComparision

****

**Figure12. Histogram of the expression values of all six patients**

From the histogram, we can say that the expression values are relatively in a similar expression pattern, but normalizing it will increase the quality of the data, so we normalize it with quartile normalization.

>petientsNormalized<-normalize.quantiles(as.matrix(petients$value),copy=TRUE)

>petients$value<-petientsNormalized

>histComparissionNormalized<-ggplot(petients, aes(value,colour=genes)) + geom\_density()

>histComparissionNormalized

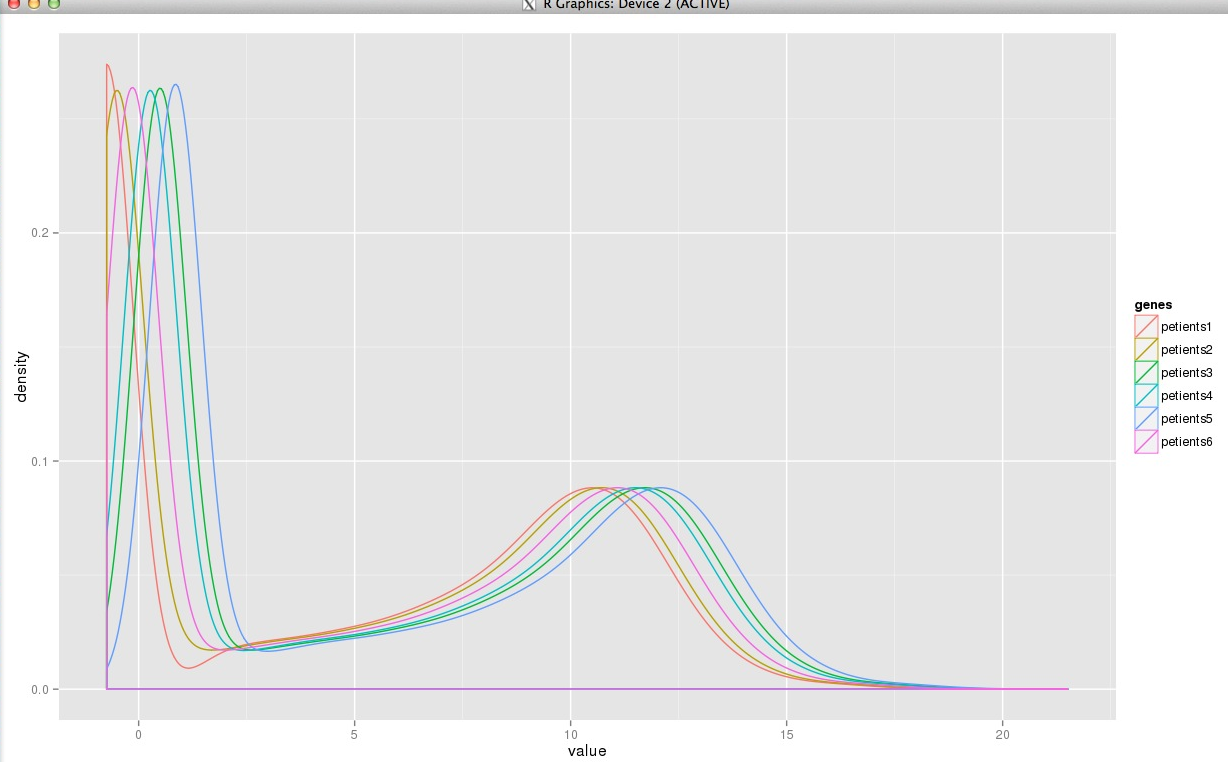


Figure 13. Histogram after normalization

As we can see from the figure 13, normalizing does not that much affected the expression patterns of the patients.

Checking the expression value of ERG gene we can see that patient 1 and patient 6 shows elevated expression level.

>ERGexpressionLevele<-as.matrix(petients[(petients$gene=="ERG"),3])

>colnames(ERGexpressionLevele)<-"ERGEspression"

>rownames(ERGexpressionLevele)<-c("petient1","petient2","petient3","petient4","petient5","petient6")

> ERGexpressionLevele

ERGEspression

petient1 14.39

petient2 10.55

petient3 10.48

petient4 10.87

petient5 16.00

petient6 9.96

The normal expression level for ERG gene is around 10. Therefore we extract patients with expression values grater than 13.

>ERGElivatedExpresedPeteints<- ERGexpressionLevele[ERGexpressionLevele[,1]>13,]

>ERGfamilyGene<-petients[grep("ERG",petients$gene),1]

**> ERGElivatedExpresedPeteints**

**petient1 petient5**

**14.39 16.00**

Then we get patient 1 and patient 6 has shown over expression of ERT. When we compare the overexpressed expression values with the normal expression values, for patient 1 it is it 4.39 over expressed in a logarithmic scale while for patient 5, it is overexpressed by logarithmic unit of 6.

Looking the PEA3 subfamily of EST family, which is ETV1, ETV4, ETV5, we can find the expression values of each of the three genes from the six patients with the following for loop:

PEA3SubFamily<-c("ETV1","ETV4","ETV5")

PEA3SubFamilyExpressionLevele<-data.frame(data=NA)

for(i in 1:length(PEA3SubFamily)){

PEA3SubFamilyExpressionLevele<-cbind(PEA3SubFamilyExpressionLevele,petients[petients$gene==PEA3SubFamily[i],3])

}

Removing ‘na’ from the PEA3SubFamilyExpressionLevele data frame:

PEA3SubFamilyExpressionLevele<-subset(PEA3SubFamilyExpressionLevele, select=-(data))

Now we can filter the expression values of PEA3 subfamily whose expression value is greater than 13 using the for loop:

>colnames(PEA3SubFamilyExpressionLevele)<-PEA3SubFamily

>rownames(PEA3SubFamilyExpressionLevele)<-c("petient1","petient2","petient3","petient4","petient5","petient6")

> PEA3SubFamilyExpressionLevele

ETV1 ETV4 ETV5

petient1 7.10 4.53 7.44

petient2 8.76 6.91 9.20

petient3 10.28 8.18 11.03

petient4 10.74 8.97 11.54

petient5 10.71 8.36 10.52

petient6 16.63 6.67 9.63

>overEXpressedPEA3FamilySample<-c()

>for(j in 1:ncol(PEA3SubFamilyExpressionLevele)){

overEXpressedPEA3FamilySample<-c(overEXpressedPEA3Family,rownames(PEA3SubFamilyExpressionLevele[PEA3SubFamilyExpressionLevele[,j]>13,]))

}

> overEXpressedPEA3FamilySample

[1] "petient6"

> PEA3SubFamilyExpressionLevele["petient6",]

ETV1 ETV4 ETV5

petient6 16.63 6.67 9.63

Again taking 13 as a threshold value only patient 6 is showing an over expression of the PEA3 subfamilies at ETV1 gene. When we compare the overexpression value with the normal expression values, which is around 10 in a logarithmic scale, it shows over expression by 6.63 logarithmic units.

**2. RNA-seq data analysis**

**Task 8. Fusion gene detection**

We read the two fasta files containing exons of TMPRSS2 and ETV1 and we find out all possible combinations of TMPRSS2 and ETV1 exons. I actually did a slight modification on the given R codes making it to find all the possible exon-exon combinations.

build\_fusion\_junctions <- function() {

outputFile <- file('TMPRSS2\_ETV\_junctions.fa', open = 'wt')

library(seqinr)

tmprss2 <- read.fasta('tmprss2\_exons.fa');

etv1 <- read.fasta('etv1\_exons.fa');

# Add a loop here to go through all combinations of exons.

left\_seq\_header<-c()

left\_seq<-list()

for(i in 1:length(tmprss2)){

left\_seq\_header[i] <- attr(tmprss2[[i]], 'name')

left\_seq[[i]]<-as.vector(tmprss2[[i]])

if (length(left\_seq[[i]])> 80) {

left\_seq[[i]] = left\_seq[[i]][(length(left\_seq[[i]])-79):length(left\_seq[[i]])]

}

}

right\_seq<-list()

right\_seq\_header<-c()

for(i in 1:length(etv1)){

right\_seq[[i]] = as.vector(etv1[[i]]);

right\_seq\_header[i] <- attr(etv1[[i]], 'name')

if (length(right\_seq[[i]]) > 80) {

right\_seq[[i]]<-right\_seq[[i]][1:80]

}

}

for(i in 1:length(tmprss2)){

for(j in 1:length(etv1)){

writeLines(c('>',left\_seq\_header[i],'\_',right\_seq\_header[j],'\n'),con = outputFile,sep="")

writeLines(c(left\_seq[[i]], right\_seq[[j]], '\n'), con = outputFile,sep="")

}

}

close(outputFile)

}

Now we got all possible combinations of the exon exon junctions between TMPRSS2 and ETV1 and saved in a file called 'TMPRSS2\_ETV\_junctions.fa'. We can check if we got 285 junction reads in our newly created fasta file:

> result <- read.fasta('TMPRSS2\_ETV\_junctions.fa')

> length(result)

[1] 285

**Task 9 building a bowtie indexes.**

[na95542@binf myFinalProject]$ bowtie/bowtie2-build ref\_sequences/TMPRSS2\_ETV\_junctions.fa index

Running the above command, we already crated a bowtie index in a directory called index.

[na95542@binf myFinalProject]$ ls index/

index.1.bt2 index.2.bt2 index.3.bt2 index.4.bt2 index.rev.1.bt2 index.rev.2.bt2

**Task 10 aligning the reads in the rna-seq directory**

The bowtie alignment command the --min-score parameter is a minimum alignment score threshold that we get a valid alignment results. The alignment score of greater than that minimum threshold is considered to be a valid alignment.

--min-score is a function governing the minimum alignment score needed for an alignment to be considered "valid". For example if I used --min-score L,0,-0.1 this function is defended as:

f(x) = 0 + -0.1 \* x, where x is a read length

therefore, for read length of 90 bases:

f(x) = 0 + -0.1 \* 90=-9

As a result in 90 base pairs of alignment, minimum score of -9 is required to consider that alignment is valid alignment.

If we consider the mismatch penalty is -1, then up to 9-mismatched alignment is considered to be valid alignment.

**Task 10. Aligning the exons of the two patients given in rna-seq directory**

Aligning the first patient the alignment result shows as below:

[na95542@binf myFinalProject]$ bowtie/bowtie2 --min-score L,0,0,-x index/index -1 rna- seq/CRPC\_543\_1.fq -2 rna-seq/CRPC\_543\_2.fq -s alignment

21536 reads; of these:

21536 (100.00%) were paired; of these:

21536 (100.00%) aligned concordantly 0 times

0 (0.00%) aligned concordantly exactly 1 time

0 (0.00%) aligned concordantly >1 times

----

21536 pairs aligned concordantly 0 times; of these:

0 (0.00%) aligned discordantly 1 time

----

21536 pairs aligned 0 times concordantly or discordantly; of these:

43072 mates make up the pairs; of these:

43072 (100.00%) aligned 0 times

0 (0.00%) aligned exactly 1 time

0 (0.00%) aligned >1 times

0.00% overall alignment rate

From the alignment result we can see that there is 0% alignment with the reference showing that the first patients does not have ETV1 fused exons. Then lets check the alignment result of the second patient:

Now lets align the second patients read in the rna-seq directory:

>bowtie/bowtie2 --min-score L,0,0,-x index/index -1 rna-seq/PCaP\_470\_1.fq -2 rna-seq/PCaP\_470\_2.fq –s alignment

2508 reads; of these:

2508 (100.00%) were paired; of these:

2508 (100.00%) aligned concordantly 0 times

0 (0.00%) aligned concordantly exactly 1 time

0 (0.00%) aligned concordantly >1 times

----

2508 pairs aligned concordantly 0 times; of these:

0 (0.00%) aligned discordantly 1 time

----

2508 pairs aligned 0 times concordantly or discordantly; of these:

5016 mates make up the pairs; of these:

5012 (99.92%) aligned 0 times

4 (0.08%) aligned exactly 1 time

0 (0.00%) aligned >1 times

0.08% overall alignment rate

From the alignment result we can see that at least there are 4 reads that have aligned exactly one time which accounts for 0.o8% of over all alignment rate. This result shows that the patient 2 or reads in CRPC\_543\_2.fq file has TMPRSS2 exon 1 and ETV1 exon 11 fusion. Exon 1 of the TMPRSS2 gene is located on the chromosome 21 and exon 11 of ETV1 is located in on chromosome 7.

1. **CHIP-Seq data analysis**

**Task 13. Building a bowtie index of the chromosome 21.**

We first build the index for our reference sequence, which is chromosome 21.

[na95542@binf myFinalProject]$ bowtie/bowtie2-build ref\_sequences/chr21.fa indexedCh21

**Task 14. Aligning chip-seq against reference sequence**

bowtie/bowtie2 indexedCh21 chip-seq/LNCaP\_R1881\_AR\_chipseq.fq -S chip\_seq\_alignmentResult.sam

43258 reads; of these:

43258 (100.00%) were unpaired; of these:

2934 (6.78%) aligned 0 times

40205 (92.94%) aligned exactly 1 time

119 (0.28%) aligned >1 times

93.22% overall alignment rate

We change the sam file to bam file and then we sort and index the bam file so that we can see the alignment using the IGV.

Changing the sam to bam file:

[na95542@binf myFinalProject]$ samtools view -bS chip\_seq\_alignmentResult.sam > chip\_seq\_alignmentResult.bam

Sorting the bam file:

[na95542@binf myFinalProject]$ samtools/samtools sort chip\_seq\_alignmentResult.bam chip\_seq\_alignmentResult.sorted

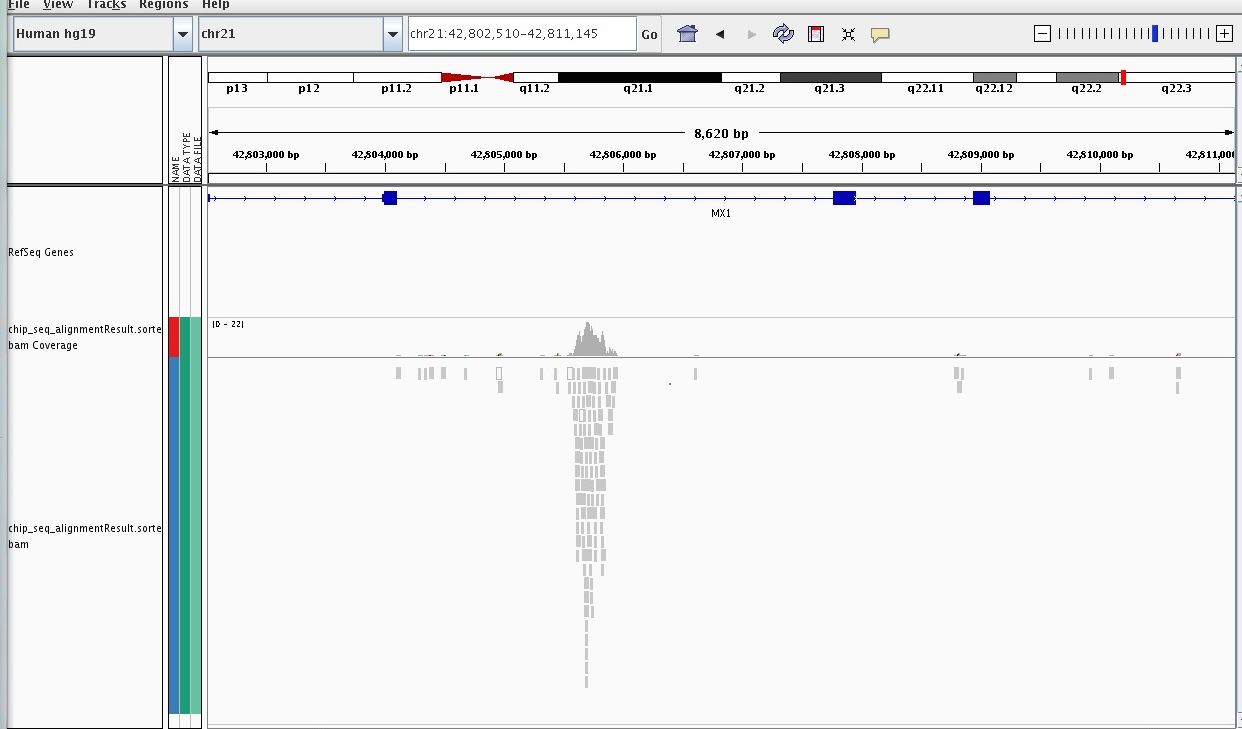
Indexing the sorted file:

[na95542@binf myFinalProject]$ samtools/samtools index chip\_seq\_alignmentResult.sorted.bam

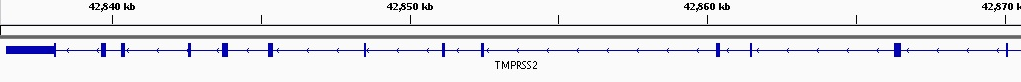
After this step we are good to visualize the alignment using IGV:

[na95542@binf myFinalProject]$ IGV\_2.1/igv.sh

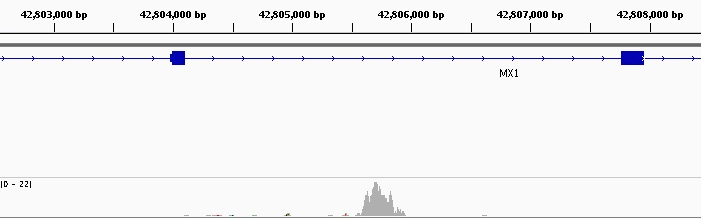
Looking the first peak from the starting point of TMPRSS2 gene,



The chromosomal coordinate at the beginning of the TMPRSS2 gene is shown around 42,840kb



We see the TSS some where around 30 kb away from the starting chromosomal coordinate of TMPRSS2 gene on the MX1 gene.



1. **Studying the mechanisms of androgen insensitivity**

Task 15. AR overexpression

Going back to our microarray expression data and looking in to the expression level of the six prostate cancer patients we got he following expression values for each of the six patients:

#Expresion level of the gene ERG

ARexpressionLevele<-as.matrix(petients[(petients$gene=="ERG"),3])

colnames(ARexpressionLevele)<-"ARExpression"

rownames(ARexpressionLevele)<-c("patient1","patient2","patient3","patient4","patient5","patient6")

> ARexpressionLevele

ARExpression

patient1 14.39

patient2 10.55

patient3 10.48

patient4 10.87

patient5 16.00

patient6 9.96

Petients1 (CRPC\_278) and patient2 (CRPC\_543) are those samples that are afflicted with castration resistant prostate

Cancers (CRPC). From the expression values of AR gene, we can see that patient 1(CRPC\_278) or the one with “CRPC\_278”file name is showing a dramatic over expression of AR gene expression.

**Task 16. Identifying the cause for the second patient with castration resistant prostate cancer**

To find out the possible reason for the cause of the second patient, we align the rna-seq reads this patient against the X chromosome. To do this, first we need to build a bowtie index of chromosome X:

[na95542@binf myFinalProject]$ bowtie2-build ref\_sequences/chrX.fa chrXIndex/chrXIndex

Then aligning using a tophat:

[na95542@binf myFinalProject]$ tophat chrXIndex/chrXIndex rna-seq/CRPC\_543\_1.fq rna-seq/CRPC\_543\_2.fq

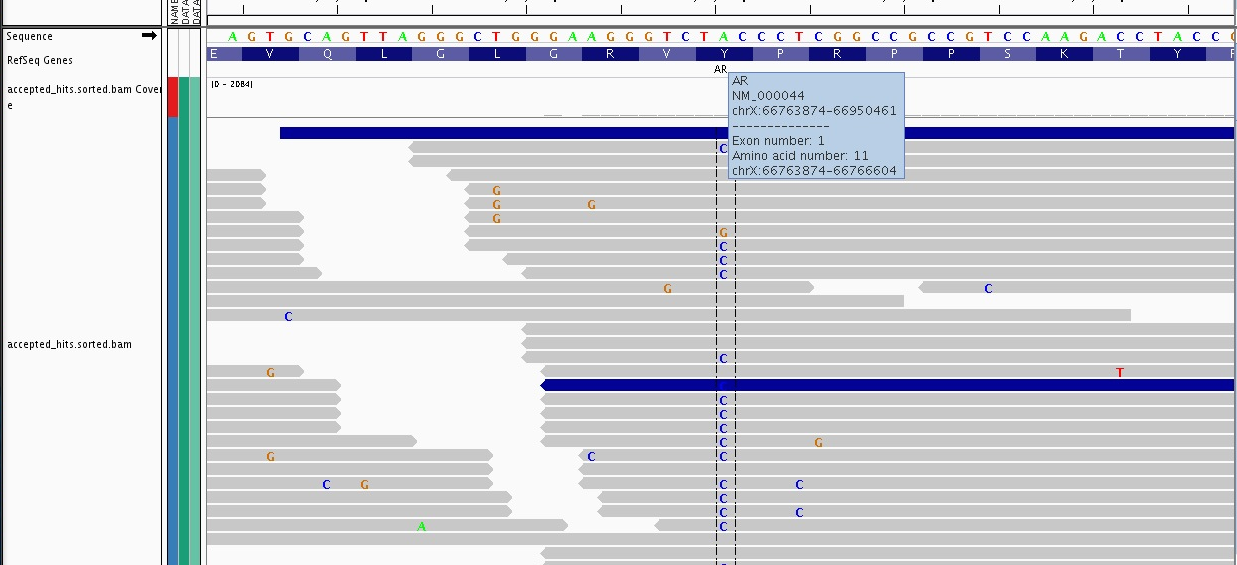
Now we got the alignment result in the tophat-output directory. Next we sort and index the accepted alignment bam file from the tophat-output directory:

[na95542@binf myFinalProject]$ samtools sort tophat\_out/accepted\_hits.bam tophat\_out/accepted\_hits.sorted

Indexing:

[na95542@binf myFinalProject]$ samtools index tophat\_out/accepted\_hits.sorted.bam

Then finally, let’s open the sorted bam file using IGV to see the mutation on the protein coding sequence of the 11th amino acid.



In total there are 27 reads with cytosine(C) and 30 reads with Adenine (A). There is also one read with Guanine (G). The ratio between the Cytosine to Adenine is 27:30, which is approximately 1:1 ratio.

The codon sequence before the mutation was ‘TAC’, which is coding for tyrosine amino acid. After the mutation, the codon sequence was changed to TCC, which is coding to serine amino acid. This amino acid change will result change in the conformation of AR protein, this in turn will permanently activates the AR protein regardless of the expression level of androgen.

1. **Appendixes**

The separate file will be attached.