Analysis of LUAD Data for Prognostic Signatures

Submitted By: **Harpreet Kaur, PhD**

Email: hks04180@gmail.com

Introduction

In this analysis, I have used TCGA-LUAD data to identify prognostic signatures. Here, I have used clinical and transcriptomics Data.

1. Data Download & Preparation of Data Matrix

Data Download

Let's downloaded TCGA-LUAD data using the `TCGAbiolinks`

```
#install.package
if (!requireNamespace("BiocManager", quietly = TRUE))
install.packages("BiocManager")
BiocManager::install("TCGAbiolinks")
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

'getOption("repos")' replaces Bioconductor standard repositories, see
'?repositories' for details

replacement repositories:
    CRAN: https://cran.rstudio.com


Bioconductor version 3.13 (BiocManager 1.30.16), R 4.1.1 (2021-08-10)

Installing package(s) 'BiocVersion', 'TCGAbiolinks'

also installing the dependencies ‘plogr’, ‘png’, ‘Biostrings’, ‘bitops’, ‘RSQLite’, ‘KE


Old packages: 'htmltools', 'knitr', 'openssl', 'pkgload', 'rmarkdown',
  'roxygen2', 'stringi', 'tibble', 'vroom', 'xfun', 'nlme'
```

```
#Load library
```

```
library("TCGAbiolinks")
```

Data Download

```
# Create Query
query_TCGA_LUAD <- GDCquery(project = "TCGA-LUAD", data.category = "Transcriptome Profiling",

# look what type of information there in data

LUAD_res = getResults(query_TCGA_LUAD)

#colnames(LUAD_res)

head(LUAD_res$sample_type )
```

```
    ---------------------------------------

    o GDCquery: Searching in GDC database

    ---------------------------------------

    Genome of reference: hg38
```

```
# download data on your system
GDCdownload(query = query_TCGA_LUAD)
```

```
    Downloading data for project TCGA-LUAD

    GDCdownload will download 592 files. A total of 308.466673 MB

    Downloading as: Tue_Sep_14_20_07_54_2021.tar.gz

    Downloading: 310 MB

    oo Filtering results
```

Data preparation

After downloading Data from GDC data portal using TCGA biolinks, clinical, metadata were downloaded directly from the portal. Data matrix files were prepared in bash, since R showing memory issues. Major steps for data preparation are as following

1. Create alist of samples directories

```
cd  /GDCdata/TCGA-LUAD/harmonized/Transcriptome_Profiling/Gene_Expression_Quantification
```

```
ls >samples_list.txt
```

2. bash script written to extract FPKM value from each sample and perform log scale transformation

   log_transform.sh:

```
while read line
do
#unzip file
gzip -d $line/*.gz

#Extract genes in a file
awk '{print $1}' $line/*FPKM.txt |perl -pe "s/\n/,/g" |perl -pe "s/,\$/\n/g"  >ger

#Extract FPKM value and take log of the value; Note, 1 was added to handle zero va
awk '{print log($2+1)}' $line/*FPKM.txt |perl -pe "s/\n/,/g" |perl -pe "s/,\$/\n/g
```

```
#provide list of sample folders
done <samples_list.txt
```

3. Run log_transform.sh script to gene expression matrix and gene list

```
sh  log_transform.sh
```

4. add genes IDs to the list

```
cat gene_list.txt expression.csv  >Gene_expression_matrix.txt
```

5. metadata.json file converted to csv file. Metadata file contains both TCGA barcode ID and file ID. Since, in our data, we have only ID. Thus, we extracted TCGA barcode IDs corresponding to file ID from the metadata file

```
cut -f3,33,34 metadata.cart.2021-09-12.csv >tcga_file_ids
cut -d- -f1-3 tcga_file_ids >tcga_3ids
cut -d- -f1-4 tcga_file_ids >tcga_4ids
paste tcga_4ids tcga_3ids tcga_file_ids >final_tcga_ids
```

6. Clinical information was extarcted from clinical matched with barcode Ids

```
cut -f2,4,10-12,15,16,48,128 clinical.tsv  >clin
cut -f2,12 exposure.tsv  >smoking_ids
sort clinical.tsv |uniq >uniq_clin.txt
```

7. add sample id to sample_list.txt

```
paste samples_list.txt Gene_expression_matrix.txt >final_gene_expression.txt
```

8. Match TCGA IDs with file IDs and keep TCGA barcode IDs as ID column in gene expression file

```
perl row_match.pl  final_gene_expression.txt final_tcga_ids 1 5 |cut -f1-2,8 >fina
```

9. Extract matched clinical data for patients

```
perl row_match.pl final_clinical.txt final_tcga_ids 1 2|cut -f1-2,8- >final_clinic
```

10. Combine clinical/metadata data with gene expression data to create a complete matrix, where first 14 columns contains clinical data and next 60,483 columns contain gene expression. Here, genes are in the columns and samples are present in rows

```
perl row_match.pl final_LUAD_gene_expression.csv final_clinical_LUAD.txt 1 1 |cut -

row_match.pl is script written in perl to match common rows from 2 files
it can be accessed here: https://github.com/pine-bio-support/ML-Demo/blob/master/rc
```

Note: Few steps (some name substitutions (with bash substitution command) performed within file in bash, those steps might not be mentioned above)

2. Load Data

Load Dataset

Here, we are uploading LUAD gene expression data with clinical data. we already prepared this file in bash after downloading data In the Clinical data, we have information regarding TCGA ids, tumor stage, vital status, vital follow-up/days to death (vital.time), geneder, race, smoking data (cigraettes per day) In the gene expression, we have 60,483 RNA transcripts for 505 patient's samples

```
#load Data
full_data <- read.table("/content/final_clinical_gene_expression_combined_LUAD.csv", sep=",",

#Check dimensions of data
dim(full_data)
```

505 · 60496

```
#View top rows
head(full_data,5)
```

	id3	tissue_type	age_at_index	ethnicity	gender	race	vital_status	vita
	<chr>	<chr>	<int>	<chr>	<chr>	<chr>	<chr>	
TCGA-05-4249-01A	TCGA-05-4249	Cancer	67	NA	male	NA	Alive	
TCGA-05-4250-01A	TCGA-05-4250	Cancer	79	NA	female	NA	Dead	
TCGA-05-4382-01A	TCGA-05-4382	Cancer	68	NA	male	NA	Alive	
TCGA-05-4384-01A	TCGA-05-4384	Cancer	66	NA	male	NA	Alive	
TCGA-05-	TCGA-05-	Cancer	70	NA	male	NA	Alive	

Extract clinical Data Extract clinical Data as clin object This is done, so we can process and gene expression data process dara properly

```
#Extract clinical Data
clin <- full_data[,1:13]

#check top 5 rows
head(clin,5)
```

A data

	id3	tissue_type	age_at_index	ethnicity	gender	race	vital_status	vita
	<chr>	<chr>	<int>	<chr>	<chr>	<chr>	<chr>	
TCGA-05-4249-01A	TCGA-05-4249	Cancer	67	NA	male	NA	Alive	
TCGA-05-	TCGA-05-	Cancer	79	NA	female	NA	Dead	

Extract expression data

Since normalization and analysis can be performed only on numeric values. Thus, we are extarcting gene expression data seprartely from clinical data

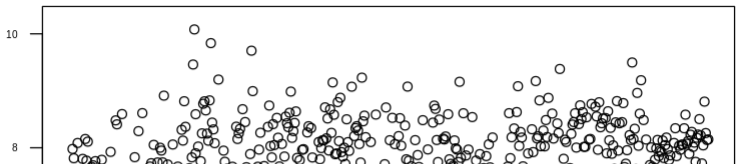
```
#Extract only gene expression Data
exp <- full_data[,14:ncol(full_data)]

#check dimensions and view top rows
dim(exp)
head(exp,5)
```

505 · 60483

ENSG00000242268.2 ENSG00000270112.3 ENSG00000167578.15 ENSG00000273842.1 EN

Double-click (or enter) to edit

```
first_500 <- exp[,1:500]

tpose_mat <- t(first_500)
boxplot(tpose_mat, cex.axis=0.5, las=2)

boxplot(first_500, cex.axis=0.5, las=2)
```



3. Exploratory Data Analysis

PCA is one of dimensionality reduction method & an exploratory data analysis method, which give us an idea regarding variability & patterns in the data.

Ideally, PCA is performed on normalized data, since, our data is already log scaled (although yet to normalize) transformed. To get an idea, we will perform PCA before and after normalization.

Here, we will perform PCA with `prcomp` functiona and subsequently, draw scatetrplot with PC1 and PC2 using `ggfortify` package

Install packages for PCA plots

```
install.packages("ggfortify")
```

```
    Installing package into ‘/usr/local/lib/R/site-library’
    (as ‘lib’ is unspecified)

    also installing the dependency ‘gridExtra’
```

Load library

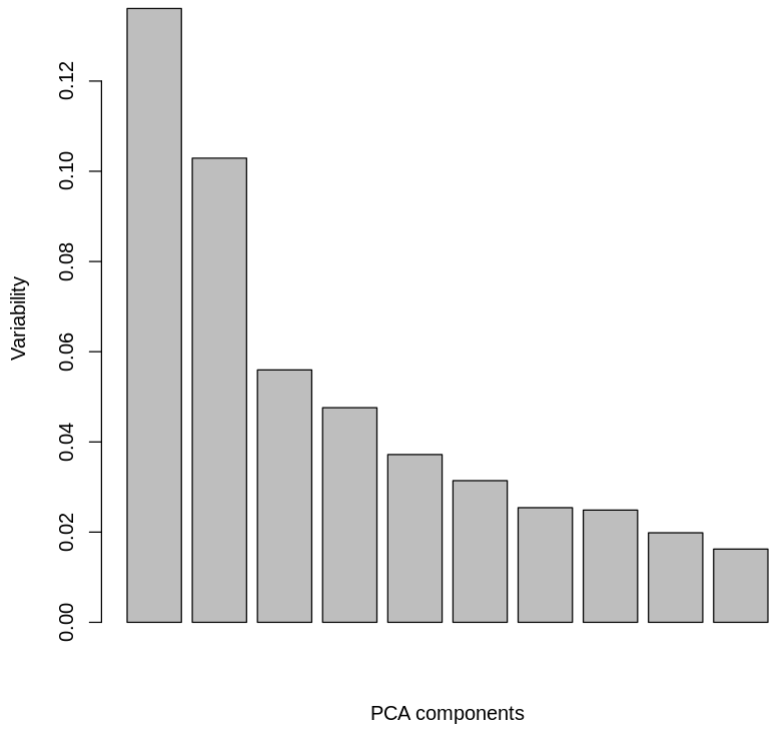
```
library(ggfortify)
```

```
    Loading required package: ggplot2
```

Create PCA objects

```
#PCA object
pca_res <- prcomp(exp)


#extract variance explained by PCA componnents
var_explained <- pca_res$sdev^2/sum(pca_res$sdev^2)

#barplot for first 10 components
```

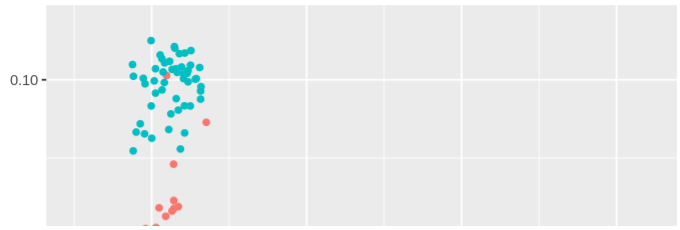
```
barplot(var_explained[1:10], xlab="PCA components", ylab="Variability")
```



Tissue type PCA plot

First, let draw PCA plot based on tissue type, i.e. Cancer & Normal. Here, we will assign color using the `colour= tissue_type` information available in our clinical data/full data

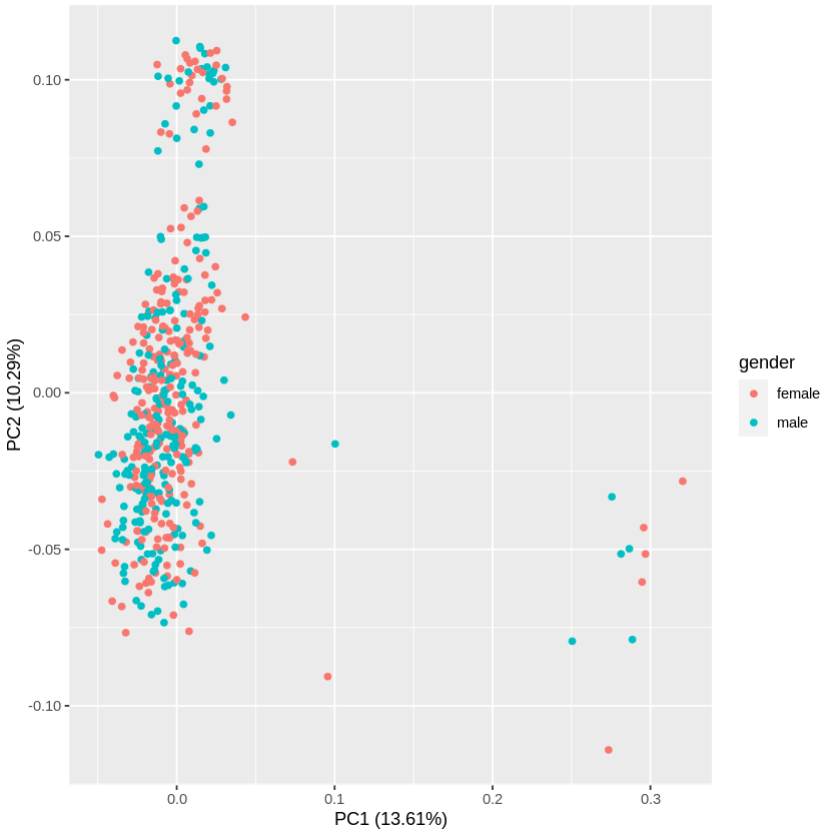
```
autoplot(pca_res, data=full_data, colour="tissue_type")
```



The PCA plot based on Tissue type indicates the clear distinction between Cancer and Normal samples. Besides, it also showing some of the outliers, we can remove them after their identification. Since, we performed PCA on only log-transformed data, thus we need to check PCA results on normalized data as well. We perform PCA with normalized data as well to make sure we have outliers in the data (We will do it later). Before that, let's explore PCA plot for other clinical characteristics, i.e. gender, stage, vital status etc.

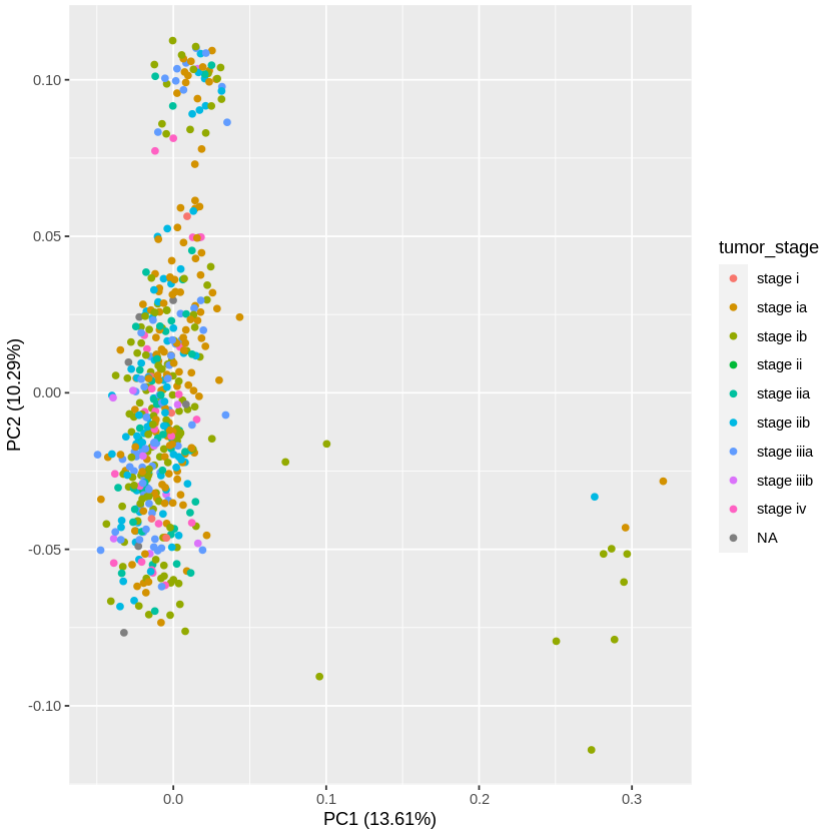
PCA plot for gender

```
autoplot(pca_res, data=full_data, colour="gender")
```



PCA plot for tumor stage

```
autoplot(pca_res, data=full_data, colour="tumor_stage")
```



PCA plot for vital status

```
autoplot(pca_res, data=full_data, colour="vital_status")
```

PCA plots have not shown any clear distict pattern for tumor stage, gender, vital status, etc.

This also indicate classifying these conditions based on molecular (transcriptomics) data is itself challenging task. Thus, we need specific molecular signatures that can stratify these groups.

Data Maniputaion

Many a times, we need to modify data/column names for analysis. Here, we are going to change some of the names in clinical data for better reprsentaion. e.g. Replacing multiple stages names with single string for stages, e.g. Stage 1A, stage 1B, Stage 1 as Stage_1

For this purpose, we will use `recode` function of the `car` package

```
#install packages
install.packages("car")
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependencies ‘matrixStats’, ‘RcppArmadillo’, ‘numDeriv’, ‘SparseM’,
```

```
#load library
library(car)
```

```
Loading required package: carData
```

Change names/value in data. Here, we will create extra columns with new names keeping original names in data

```
clin$OS <- recode(clin$vital_status,"c('Dead')=1;c('Alive')=0")
```

```
clin$stage <- recode(clin$tumor_stage,"c('stage ia','stage ib','stage i')='Stage_1';c('stage
```

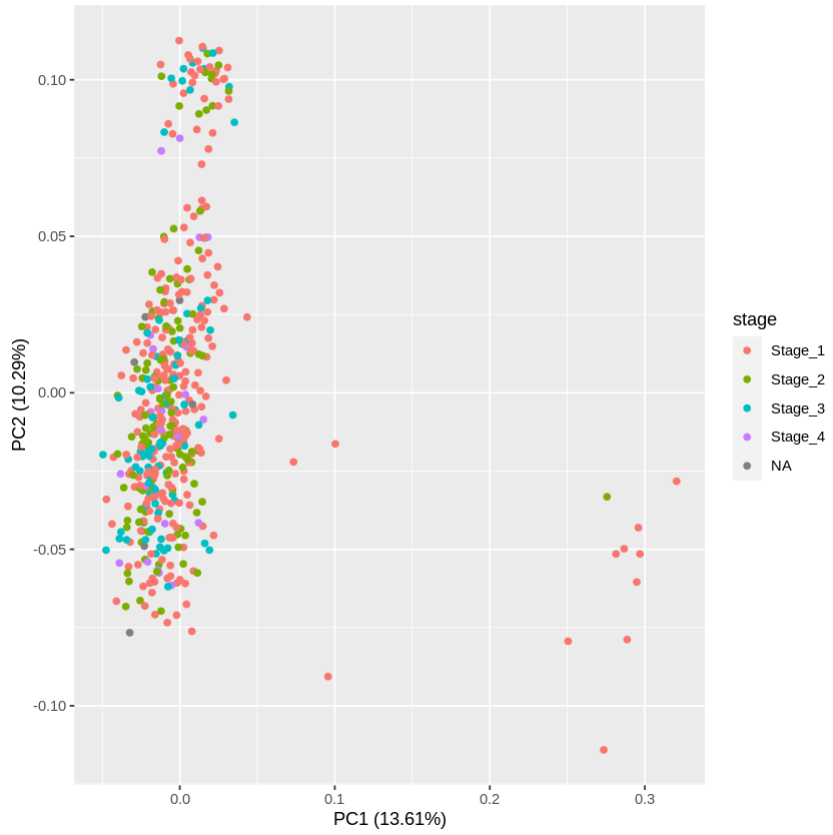
```
clin$stage_n <- recode(clin$tumor_stage,"c('stage ia','stage ib','stage i')=1;c('stage iia','
```

Combine clinical data (conatining new columns) with expression data. Here, we are combining both data column-wise using `cbind`

```
Full_data1 <- cbind(clin, exp)
```

To make sure, new column added in clinical data; let's draw PCA plot with new column `stage`

```
autoplot(pca_res, data=Full_data1, colour="stage")
```



4. Training & Test data preparation & Normalization for Analysis

We can apply machine learning and statiscal analytical methods primarily on normalized data. But, before normalization, we must prepare our data properly. Here, we will prepare our training and test data.

Training data will be used for used feature selection/feature reduction and to train the models.

Test data will be used assess the performance of prediction model devevloped based on identified molecular signatures.

Since, in this analysis, our goal is to identify prognostic signature/predictor for LUAD cancer.

From the previous literature it is known that tumor stage is the key factor which plays a vital role in the prognosis of patients. Thus, first, we will prepare training and test dataset based on tumor stage.

80% data from each stage will be included in training data and 20% data will be included in test data.

Besides, exploratory data analysis using PCA (see PCA plot based on stage) shows that there are some samples where we have no information (NA)regarding tumor stage. Thus, we considering only stage_1, stage_2 , stage_3 and stage_4 samples.

Extract data for each stage

```
# Extract data for each stage
stage1 = subset(Full_data1, stage == "Stage_1")
stage2 = subset(Full_data1, stage == "Stage_2")
stage3 = subset(Full_data1, stage == "Stage_3")
stage4 = subset(Full_data1, stage == "Stage_4")

#Check dimensions of extracted data
dim(stage1)
dim(stage2)
dim(stage3)
dim(stage4)
```

267 · 60499
122 · 60499
85 · 60499
24 · 60499

Prepare training (80%) and test (20%) data for each stage independently

```
set.seed(7)
#Split training and test data into 80:20 ratio
dt1 = sort(sample(nrow(stage1), nrow(stage1)*.8))

#Training Data
train_s1<-stage1[dt1,]

#Test Dataset
test_s1<-stage1[-dt1,]

#Split training and test data into 80:20 ratio
dt2 = sort(sample(nrow(stage2), nrow(stage2)*.8))

#Training Data
train_s2<-stage2[dt2,]
```

```
#Test Dataset
test_s2<-stage2[-dt2,]


#Split training and test data into 80:20 ratio
dt3 = sort(sample(nrow(stage3), nrow(stage3)*.8))

#Training Data
train_s3<-stage3[dt3,]

#Test Dataset
test_s3<-stage3[-dt3,]


#Split training and test data into 80:20 ratio
dt4 = sort(sample(nrow(stage4), nrow(stage4)*.8))

#Training Data
train_s4<-stage4[dt4,]

#Test Dataset
test_s4<-stage4[-dt4,]


# final train and test

train_set <- rbind(train_s1, train_s2, train_s3, train_s4)

test_set <- rbind(test_s1, test_s2, test_s3, test_s4)


#Write into files
write.table(train_set,·file·=·"train_set.txt",·sep="\t",·quote=F,·row.names·=·F)

write.table(test_set, file = "test_set.txt", sep="\t", quote=F, row.names = F)
```

Extract clinical data for training and test

```
#Extract clinical data for training and test
train_clin <- train_set[,1:16]
test_clin <- test_set[,1:16]

#View top rows
head(train_clin[1:16],5)
```

	id3	tissue_type	age_at_index	ethnicity	gender	race	vital_status	vita
	<chr>	<chr>	<int>	<chr>	<chr>	<chr>	<chr>	
TCGA-05-4249-01A	TCGA-05-4249	Cancer	67	NA	male	NA	Alive	
TCGA-05-4382-01A	TCGA-05-4382	Cancer	68	NA	male	NA	Alive	
TCGA-05-4389-01A	TCGA-05-4389	Cancer	70	NA	male	NA	Alive	
TCGA-05-4390-01A	TCGA-05-4390	Cancer	58	NA	female	NA	Alive	
TCGA-	TCGA							

Extract only expression data for training and test data (which is required for normalization

```
#extract only expression values
train_exp <- train_set[17:ncol(train_set)]
test_exp <- test_set[17:ncol(test_set)]

#Check dimensions
dim(train_exp)
dim(test_exp)
```

397 · 60483
101 · 60483

Data Pre-processing & Normalization

It is important to remove those features (genes) from data that contribute only noise Thus, we will remove those genes having zero value and equal variance in 80% of samples using `NZV` method from `caret` package and `predict` function used for predicting values for training and test data

Next, we will normalize our training and test data with center and scale methods. Eventually, we will obtain zscore normalized data

```
#install caret package
install.packages("caret")
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependencies ‘listenv’, ‘parallelly’, ‘future’, ‘globals’, ‘future.
```

```
#load library
library("caret")
```

```
Loading required package: lattice

Warning message in system("timedatectl", intern = TRUE):
“running command 'timedatectl' had status 1”
```

```
# remove near zero variation for the columns at least or 80% of the values are the same
# this function creates the filter
nzv <- preProcess(train_exp,method="nzv",uniqueCut = 20)

# apply the above created filter using "predict" function
nzv_exp_tr <- predict(nzv,train_exp)
nzv_exp_te <- predict(nzv,test_exp)

# center & scaling
processCenter <- preProcess(nzv_exp_tr, method = c("center", "scale"))

# apply center & scaling on traning and test data
Norm_tr_exp <- predict(processCenter,nzv_exp_tr)
Norm_te_exp <- predict(processCenter,nzv_exp_te)

#round up value upto 3 digits
Norm_tr_exp <- round(Norm_tr_exp,3)
Norm_te_exp <- round(Norm_te_exp,3)
```

```
#check dimension of normalized data
dim(Norm_tr_exp)
dim(Norm_te_exp)
```

```
397 · 39482
101 · 39482
```

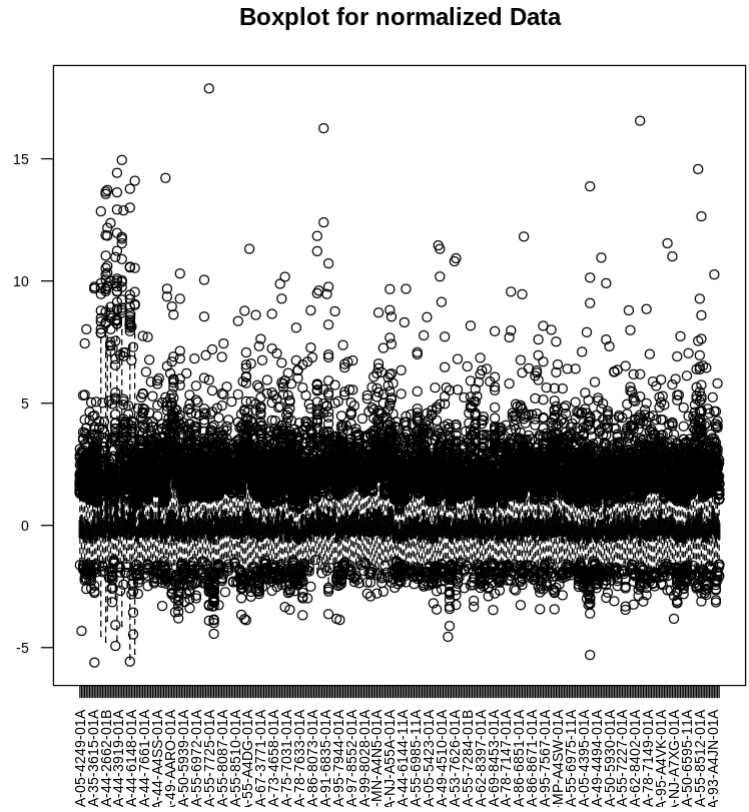
```
train_set1 <- cbind(train_clin, Norm_tr_exp )
test_set1 <- cbind(test_clin, Norm_te_exp )
```

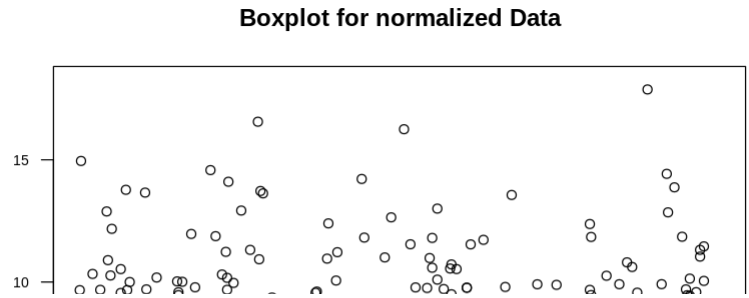
Visualization of normalized data

```
head(Norm_tr_exp[1:10],5)
```

	ENSG00000242268.2	ENSG00000270112.3	ENSG00000167578.15	ENSG00000078237.5	EN
	<dbl>	<dbl>	<dbl>	<dbl>	
TCGA-05-4249-01A	-0.157	-0.021	-0.903	-0.293	
TCGA-05-4382-01A	-0.362	-0.222	-0.071	0.774	
TCGA-05-4389-01A	-0.481	0.051	0.010	0.991	
TCGA-05-4390-01A	-0.481	-0.089	-1.316	0.382	
TCGA-05-4420-01A	-0.481	-0.172	-1.475	1.784	

```
### Visulization
Norm_exp_t <- t(Norm_tr_exp[1:500])

boxplot(Norm_exp_t, las=2, cex.axis=0.7, main="Boxplot for normalized Data")

boxplot(Norm_tr_exp[1:500], las=2, cex.axis=0.7, main="Boxplot for normalized Data")
```

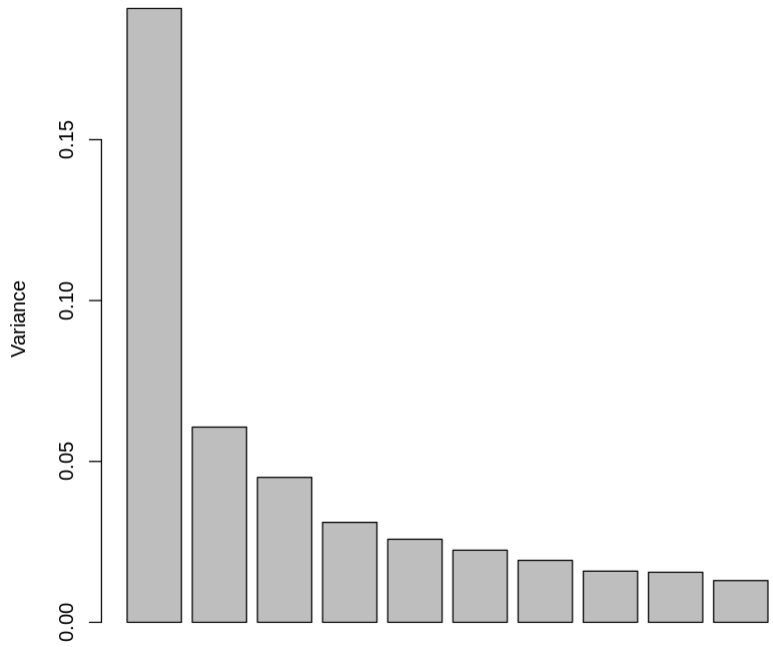




5. Exploratory Data analysis after Data Normalization**

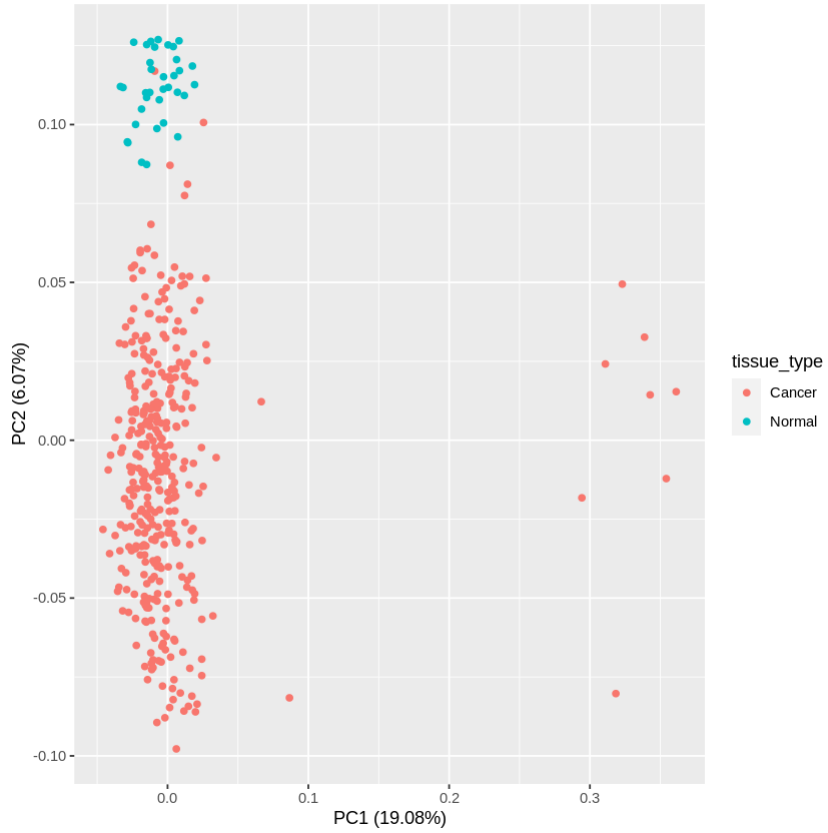
Create PCA object for normalized Data

```
#PCA for pre-processed data
pca_res1 <- prcomp(Norm_tr_exp)

#variance explained by PCA components
var_explained1 <- pca_res1$sdev^2/sum(pca_res1$sdev^2)

#barplot for first 10 components
barplot(var_explained1[1:10], xlab="PCA components", ylab="Variance")
```



PCA plot for tissue type

```
autoplot(pca_res1, data=train_set, colour="tissue_type")
```

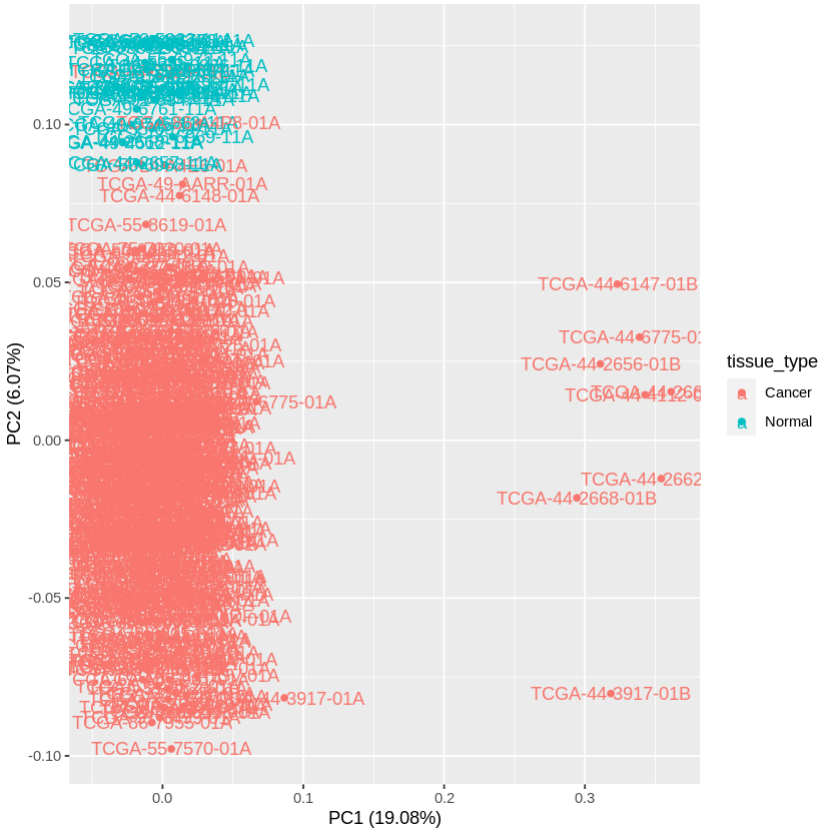


PCA plot based on normalized data showing the better separation and PC1 and PC2 reprsents more variability, i.e., ~25%. But, still it showing outliers (on the right side). Next, let's identify these outliers

Outliers Identification

To identify outleir samples, we can label samples using `label=TRUE`

```
autoplot(pca_res1, data=train_set, colour="tissue_type", label=TRUE)
```



On assessing these outliers, it has been observed that these samples ID names ends with -01B or -01C; which represent different vials for samples from the same site.

This also indicate, handling of data extraction or sample extraction can impact the analysis

Outliers Removal

Find the index (row number) for outliers

To Remove the outliers , next, we will identify the index (row number) with sample ID using `which` function

```
#get row index
which(rownames(train_set1) %in% c("TCGA-44-3917-01B", "TCGA-44-2668-01B", "TCGA-44-2662-01B",
```

14 · 17 · 18 · 20 · 24 · 27 · 32 · 35 · 87 · 241

Outliers Removal

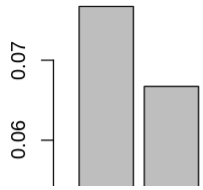
Here, we will remove outlier samples using index values

```
# drop row with row index
train_set2 <- train_set1[-c(14, 17, 18, 20, 24, 27, 32,35, 87, 241),]
dim(train_set1)
dim(train_set2)
```

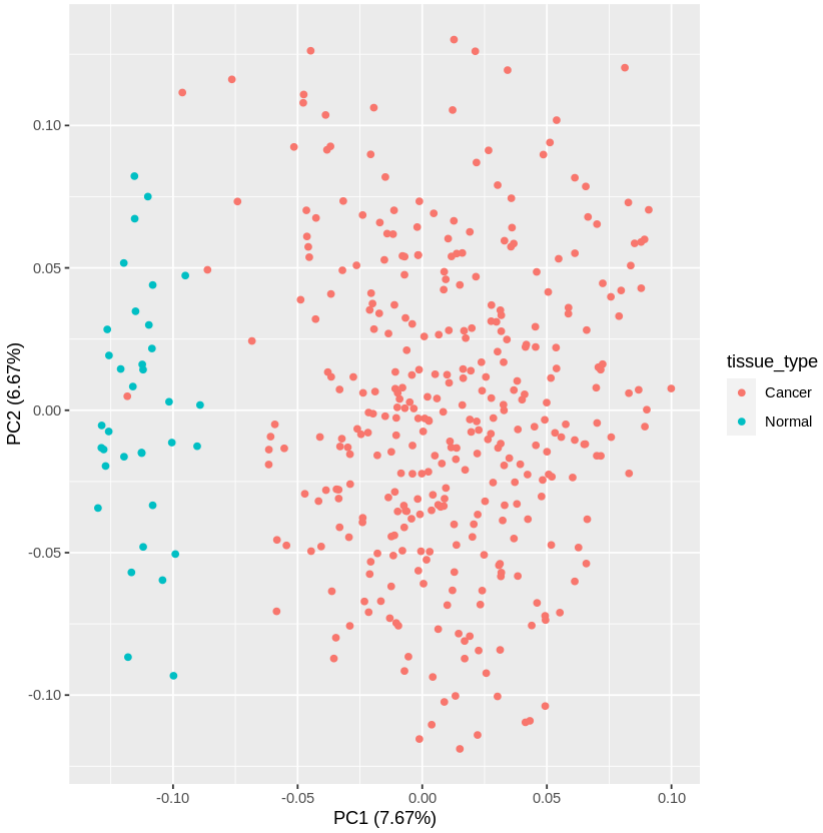
397 · 39498
387 · 39498

PCA after outlier removal

```
data_pca <- train_set2[17:ncol(train_set2)]


#PCA for pre-processed data
pca_res2 <- prcomp(data_pca)

#variance explained by PCA components
var_explained2 <- pca_res2$sdev^2/sum(pca_res2$sdev^2)

#barplot for first 10 components
barplot(var_explained2[1:10], xlab="PCA components", ylab="Variance")
```




```
autoplot(pca_res2, data=train_set2, colour="tissue_type")
```



After outliers removal, we can clearly see the distict pattern in cancer and normal samples Now, our training data is ready for downstream analysis.

```
dim(train_set2)
```

387 · 39498

6. Feature selection/reduction

In our training data, we still have more than 39K features or RNA transcripts. To develop a good prediction model, we must must have only important features, other will results only in noise. Besides, ML prediction models based on large number of features usually computationally very expensive.

Thus, it is important to consider only relevant features in ML based biomedical data analysis. There are number of approaches that can be used to identify relevant features.

Let's first apply simplest approach, i.e. significantly differentially expressed genes between cancer and normal or non-tumorous samples. Since, we are looking for prognostic signatures for LUAD patients, thus genes that similarly expressed between cancer and normal definitly not relevant.

Differential Gene Expression Analysis

There are number of methods, i.e. Wilcoxon T-test, DESeq2, EdgeR, Limma that can be used to identify differentially expressed genes. Choice of method would be based on type of values in the data. For instance, DESeq2 and EdgeR can be applied on non-normalized data Raw read count data, Limma can used for microarray data. Wilcoxon and Welch tests can applied on normalized data.Since, our data is already Zscore- normalized , thus we can apply directly wilcoxon ranking test.

Prepare data for Wilcoxon ranking T-test

Since, we want to identify significantly differentially expressed genes between cancer and normal samples. Now, let's extract data for cancer and normal samples. We added Cancer and normal information based on TCGA barcode ID.

```
#Prepare groups
group1 = subset(train_set2, tissue_type== "Cancer")
group2 = subset(train_set2, tissue_type== "Normal")

dim(group1)
dim(group2)
```

352 · 39498
35 · 39498

```
head(group1[17:20],5)
```

A data.frame: 5 × 4

	ENSG00000242268.2	ENSG00000270112.3	ENSG00000167578.15	ENSG00000078237.5
	<dbl>	<dbl>	<dbl>	<dbl>

```
#combine cancer & Normal
C_N_data <- rbind(group1[17:ncol(group1)], group2[17:ncol(group2)])

#prepare t_test input
t_test_input <- t(C_N_data)

dim(t_test_input)

#view first 5 rows of the data
head(t_test_input,5)

#Extract feature names
genes <- row.names(t_test_input)
```

39482 · 387

A matrix: 5 × 387 of

	TCGA-05-4249-01A	TCGA-05-4382-01A	TCGA-05-4389-01A	TCGA-05-4390-01A	TCGA-05-4420-01A	TCGA-05-4426-01A	TCGA-05-4433-01A	TCGA-05-5715-01A	TCGA-35-3615-01A	TC 46
ENSG00000242268.2	-0.157	-0.362	-0.481	-0.481	-0.481	-0.481	-0.149	-0.481	-0.481	-0.
ENSG00000270112.3	-0.021	-0.222	0.051	-0.089	-0.172	-0.122	-0.222	-0.073	-0.222	0.
ENSG00000167578.15	-0.903	-0.071	0.010	-1.316	-1.475	-1.879	-0.036	-0.988	0.130	-0.
ENSG00000078237.5	-0.293	0.774	0.991	0.382	1.784	-0.556	0.006	0.372	0.663	-0.
ENSG00000146083.10	0.623	-0.714	-0.372	-0.234	-0.810	2.922	0.658	-1.522	0.829	2.

```
dim(t_test_input)
```

39482 · 387

Wilcoxon Test

Apply Wilcoxon test on data between cancer & Normal samples

```
Wilcoxon <- apply(t_test_input[,1:ncol(t_test_input)], 1, function (x) wilcox.test(x[1:352],›
```

Extract p-values, FDR, Fold change(FC), Bonferroni adjusted p-value from the results

```
#Extract P-values
p_value_w <- unlist(lapply(Wilcoxon, function(x) x$p.value))

#Extract FDR
fdr_w <- p.adjust(p_value_w, method = "fdr")

#Extract Boneferroni adjusted P-values
boneferroni_w <- p.adjust(p_value_w, method = "bonferroni")

#Make list
list_w <- Map(c, p_value_w, fdr_w, boneferroni_w)

#make dataframe of the list
result_w<- data.frame(matrix(unlist(list_w), nrow=length(list_w), byrow=T))

#calculate the mean of each gene per cancer group group
cancer = apply(t_test_input[,1:352], 1, mean)

#calcuate the mean of each gene per normal group
normal = apply(t_test_input[, 353:387], 1, mean)

foldchange <- as.data.frame(cancer/normal)

#Combine with the gene
result_w1 <-cbind(genes,result_w,foldchange )

#Provide the column names to both groups
colnames(result_w1) <- c("Gene_ID","P_value","FDR","Boneferroni", "FC")
```

Identification significantly DEGs

Identification significantly differentially expressed genes

first Sort results based on FDR value using `order` function. Here, I am using FDR < 0.05 to identify significant genes.

Further, applying Fold change criteria FC >= +1.2 and FC< = -1.2 for identification of upregulated genes and downregulated genes, respectively.

```
# Sort features based on FDRy value
sorted_features <- as.data.frame(result_w1[order(result_w1[,"FDR"], decreasing =  FALSE),])

#select significant features based on FDR <0.05
sig_features <- sorted_features[sorted_features$FDR < 0.05, ]

#select top features based on FDR <0.05 & FC > +/- 1.2
# upregulated genes
upreg_features <- sig_features[sig_features$FC > 1.2, ]
```

```
#upregulated genes
downreg_features <- sig_features[sig_features$FC < -1.2, ]

#Check dimensions
dim(upreg_features)
dim(downreg_features)

#Create top feature combining both significantly upregulated and downregulated genes
top_features <- rbind(upreg_features,downreg_features)
dim(top_features)

#Write results into a file
write.table(top_features,file="significant_gene_result.txt", sep='\t',  quote = F,row.names =

#view top rows for results
head(top_features)
```

```
263 · 5
331 · 5
594 · 5
```

A data.frame: 6 × 5

	Gene_ID	P_value	FDR	Boneferroni	FC
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>
ENSG00000250280.2	ENSG00000250280.2	1.047854e-11	8.902812e-11	4.137137e-07	251.988636
ENSG00000181541.5	ENSG00000181541.5	5.349235e-10	3.488578e-09	2.111985e-05	3.891182
ENSG00000251011.4	ENSG00000251011.4	9.916718e-10	6.227642e-09	3.915319e-05	7.346728
ENSG00000235731.1	ENSG00000235731.1	1.195481e-09	7.421380e-09	4.719997e-05	4.649381

From the Differentially Gene expression analysis, we are left with few hundered important features from 39K. Now, we can use these features for downstream analysis

Draw Volcano plot to visualize significant gene pattern

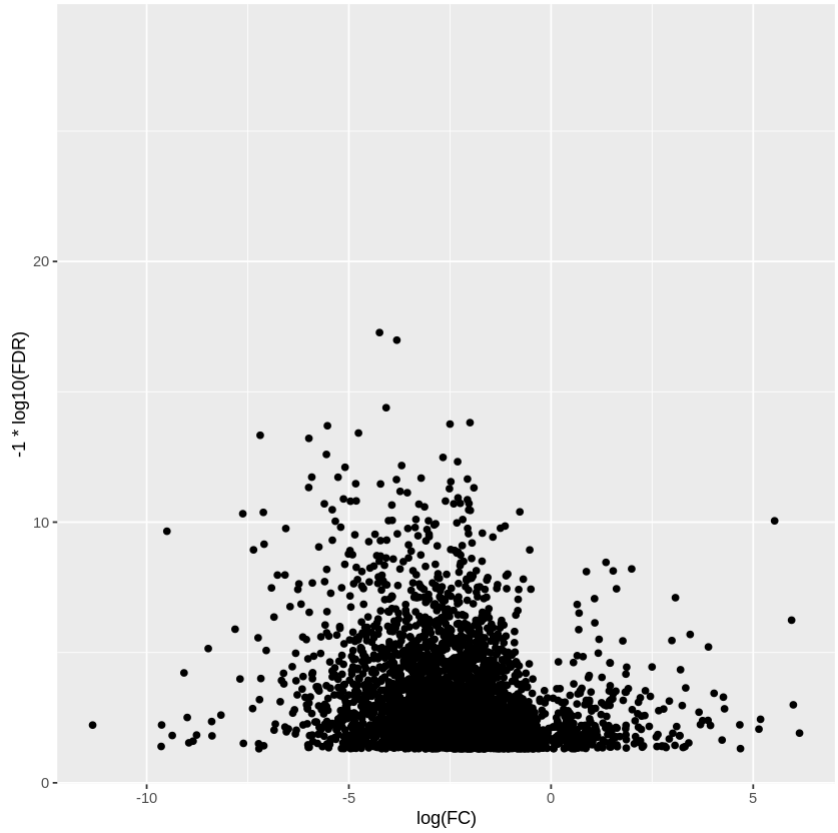
```
install.packages("ggplot2")
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

```
library(ggplot2)
```

```
#draw volcano plot
volcano = ggplot(data = sig_features, aes(x = log(FC), y = -1*log10(FDR)))
volcano + geom_point()
```

```
Warning message in log(FC):
“NaNs produced”
Warning message in log(FC):
“NaNs produced”
Warning message:
“Removed 19286 rows containing missing values (geom_point).”
```



Create Training and test data with selected features (significant genes) only

```
#Prepare dataset (training/test) for selected only
top1 <- as.data.frame(top_features$Gene_ID)
colnames(top1) <- c("ID")
#head(top1)

#Transpose training and test data to extract genes
train_set2_t <- t(train_set2)
test_set1_t <- t(test_set1)

# Prepare training data with only selected top features
tr_selected_data <- as.data.frame(train_set2_t[row.names(train_set2_t) %in% c(top1$ID), ])
```

```
# Prepare test data with only selected top features
te_selected_data <- as.data.frame(test_set1_t[row.names(test_set1_t) %in% c(top1$ID), ])

#Check dimensions of dataset
dim(tr_selected_data)
dim(te_selected_data)
```

```
594 · 387
594 · 101
```

```
head(tr_selected_data,5)
head(te_selected_data,5)
```

A data.frame: 5 ×

	TCGA-05-4249-01A	TCGA-05-4382-01A	TCGA-05-4389-01A	TCGA-05-4390-01A	TCGA-05-4420-01A	TCGA-05-4426-01A	TCGA-05-4433-01A	TCGA-05-5715-01A	TCGA-35-3615-01A	TCG 3 463 0
	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<ch
ENSG00000259883.1	-0.068	-0.353	-0.362	-0.367	-0.234	-0.290	0.491	-0.348	-0.457	-0.2
ENSG00000231981.3	-0.121	-0.157	-0.250	-0.250	-0.250	-0.250	-0.250	-0.250	-0.144	-0.2
ENSG00000263089.1	-0.085	-0.468	-0.491	-0.472	-0.320	-0.334	1.369	-0.423	-0.283	-0.4
ENSG00000233540.1	1.450	0.129	-0.040	-0.455	-0.455	-0.455	-0.455	0.431	-0.455	0.3
ENSG00000273797.1	-0.353	-0.192	-0.288	-0.417	-0.369	-0.056	-0.287	-0.276	-0.161	-0.1

A data.frame: 5 ×

	TCGA-05-4417-01A	TCGA-05-4422-01A	TCGA-35-4122-01A	TCGA-38-4625-01A	TCGA-38-4631-01A	TCGA-44-2657-01A	TCGA-44-2666-01A	TCGA-44-5645-01B	TCGA-44-7671-01A	TCG 4 767 0
	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<ch
ENSG00000259883.1	-0.089	-0.267	-0.244	-0.211	-0.522	0.063	2.793	8.409	-0.382	-0.3
ENSG00000231981.3	-0.008	-0.250	-0.250	-0.250	-0.250	-0.250	3.460	7.998	-0.250	-0.2
ENSG00000263089.1	-0.398	-0.051	-0.512	-0.458	-0.467	-0.394	1.898	6.165	-0.405	-0.3
ENSG00000233540.1	-0.455	-0.455	-0.455	0.863	-0.455	-0.455	0.931	17.132	-0.455	0.0
ENSG00000273797.1	-0.417	-0.115	-0.265	-0.331	-0.417	-0.173	1.187	6.478	-0.417	-0.2

Prepare final training and test data for selected features

```
#transpose train and test data
tr <-t(tr_selected_data)
```

```
te <-t(te_selected_data)

#check dimensions
dim(tr)
dim(te)

#Extract clinical data for train and test
clin_tr <- train_set2[1:16]
clin_te <- test_set1[1:16]
```

387 · 594
101 · 594

Transpose Data

```
# combine clinical data with gene expression for selected train & Test data
sel_tr <- cbind(clin_tr, tr)
sel_te <- cbind(clin_te, te)

#Write training and test Data into a files
write.table(sel_tr,file="Selected_train_data.txt", sep='\t',  quote = F,row.names = FALSE)
write.table(sel_te,file="Selected_test_data.txt", sep='\t',  quote = F,row.names = FALSE)

#Check dimensions
dim(sel_tr)
dim(sel_te)

#view top rows of data
head(sel_tr,5)
head(sel_te,5)
```

387 · 610
101 · 610

	id3	tissue_type	age_at_index	ethnicity	gender	race	vital_status	vita
	<chr>	<chr>	<int>	<chr>	<chr>	<chr>	<chr>	
TCGA-05-4249-01A	TCGA-05-4249	Cancer	67	NA	male	NA	Alive	
TCGA-05-4382-01A	TCGA-05-4382	Cancer	68	NA	male	NA	Alive	
TCGA-05-4389-01A	TCGA-05-4389	Cancer	70	NA	male	NA	Alive	
TCGA-05-4390-01A	TCGA-05-4390	Cancer	58	NA	female	NA	Alive	
TCGA-05-4420-01A	TCGA-05-4420	Cancer	41	NA	male	NA	Alive	

	id3	tissue_type	age_at_index	ethnicity	gender	race	vital_status	vita
	<chr>	<chr>	<int>	<chr>	<chr>	<chr>	<chr>	
TCGA-05-4417-01A	TCGA-05-4417	Cancer	51	NA	female	NA	Alive	

7. Survival Analysis

To identify prognostic factor for LUAD , now let's first perform univariate survival analysis with clinical and molecular features using `survival` and `survminer` packages

	35-	Cancer	69	hispanic or	male	white	Alive	

```
# install packages
install.packages("survminer")
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependencies ‘bitops’, ‘corrplot’, ‘markdown’, ‘RCurl’, ‘png’, ‘jpe
```

```
#Load required libraries
library(survival)
library(survminer)
#library(dplyr)
```

```
Attaching package: ‘survival’

The following object is masked _by_ ‘.GlobalEnv’:

    cancer

The following object is masked from ‘package:caret’:

    cluster

Loading required package: ggpubr

Attaching package: ‘survminer’

The following object is masked from ‘package:survival’:

    myeloma
```

Univariate Survival Analysis

Let's first perform univariate survival analysis for each of clinical and significant DEGs genes

Remove samples where OS.time or survival time information is missing

```
#remove samples where OS.time is NA
input_data_tr1<-subset(sel_tr,vital.time!="NA")

#Check dimensions of data
dim(input_data_tr1)
```

```
382 · 610
```

Create survival object

```
# create survival object
surv_object <- Surv(time = as.numeric(input_data_tr1$vital.time), event = input_data_tr1$OS)
```

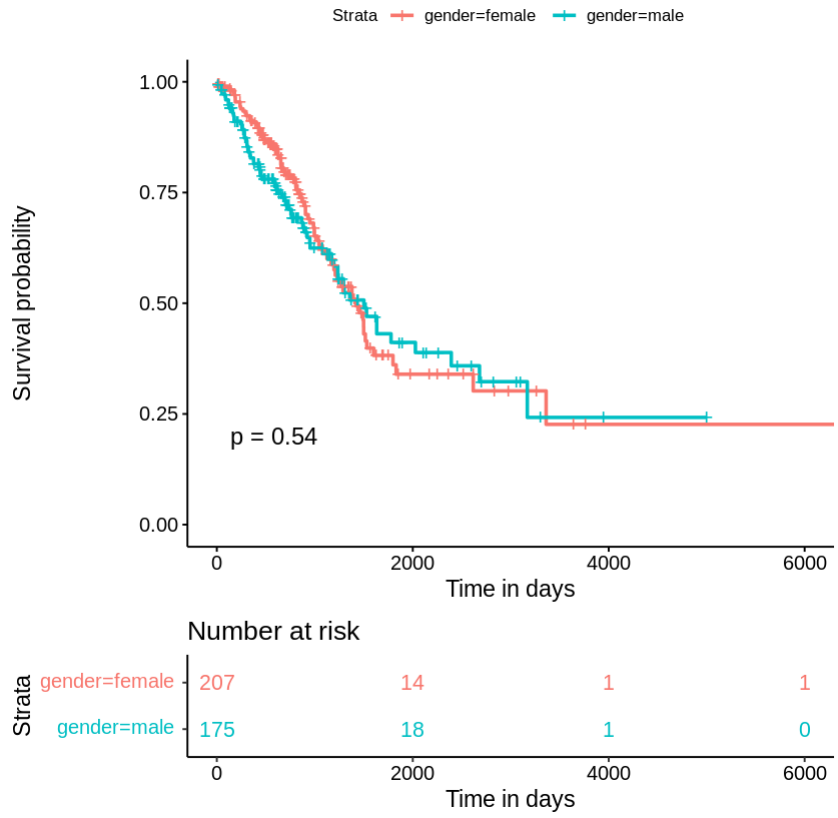
Develop survival model for Gender

```
# Gender based
fit = survfit(surv_object ~ gender, data=input_data_tr1)

print(fit)
```

```
     Call: survfit(formula = surv_object ~ gender, data = input_data_tr1)

                    n events median 0.95LCL 0.95UCL
     gender=female 207     75   1421    1197    1798
     gender=male   175     68   1501    1194    2681
```

Create kaplan-meier plot to visualize the startifcation of risk groups

```
# we produce a Kaplan Meier plot
ggsurvplot(fit, data=input_data_tr1, pval=TRUE, risk.table=TRUE, xlab="Time in days", risk.ta
```



Survival model based on Age group

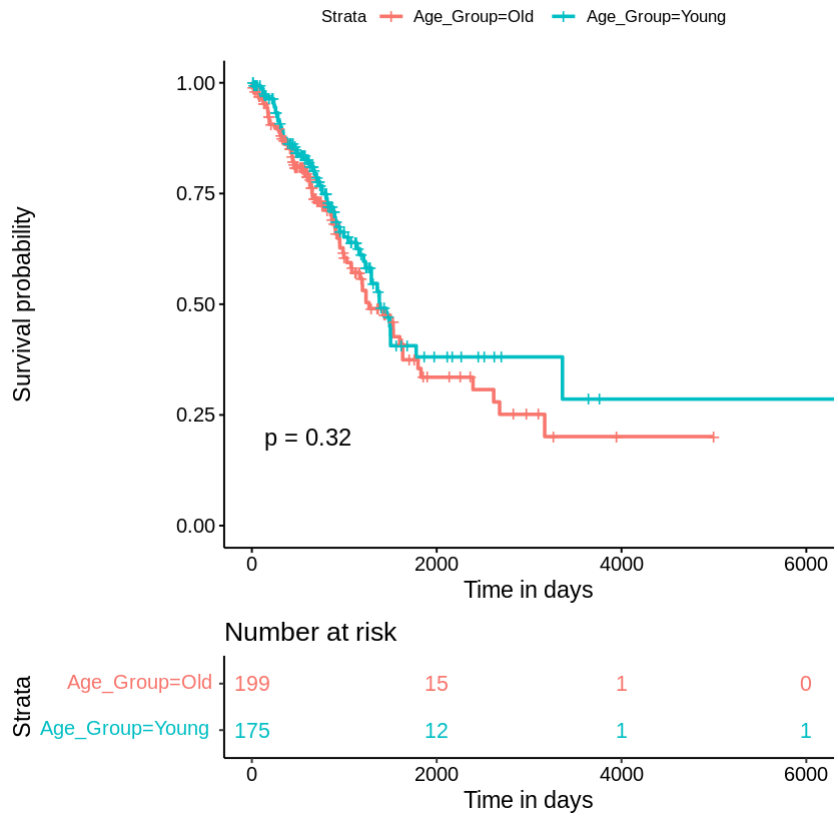
Here, we are creating two group old >65 and young < 65 years, since mean age is 65

```
#remove samples with no age information
input_data_tr3<-subset(input_data_tr1, age_at_index!="NA")

# Create age groups based of mean age (which is nearly 65)
age = as.data.frame(ifelse(input_data_tr3$age_at_index >= mean(as.numeric(input_data_tr3$age_
colnames(age) <- c("Age_Group")

#Add age group column to data
input_data_tr3 <- cbind(age,input_data_tr3)

head(input_data_tr3 )
```

	Age_Group	id3	tissue_type	age_at_index	ethnicity	gender	race	vital_s
	<chr>	<chr>	<chr>	<int>	<chr>	<chr>	<chr>	
TCGA-05-4249-01A	Old	TCGA-05-4249	Cancer	67	NA	male	NA	
TCGA-05-4382-01A	Old	TCGA-05-4382	Cancer	68	NA	male	NA	
TCGA-05-4389-01A	Old	TCGA-05-4389	Cancer	70	NA	male	NA	
TCGA-05-4390-01A	Young	TCGA-05-4390	Cancer	58	NA	female	NA	
TCGA-05-4420-01A	Young	TCGA-05-4420	Cancer	41	NA	male	NA	
TCGA-05-4426-01A	Old	TCGA-05-4426	Cancer	71	NA	male	NA	

```
#Create survival object
surv_object_a <- Surv(time = as.numeric(input_data_tr3$vital.time), event = input_data_tr3$OS

#fit model
```

```
fit_a = survfit(surv_object_a ~ Age_Group, data=input_data_tr3)

# Kaplan Meier plot
ggsurvplot(fit_a, data=input_data_tr3, pval=TRUE, risk.table=TRUE, xlab="Time in days", risk.
```

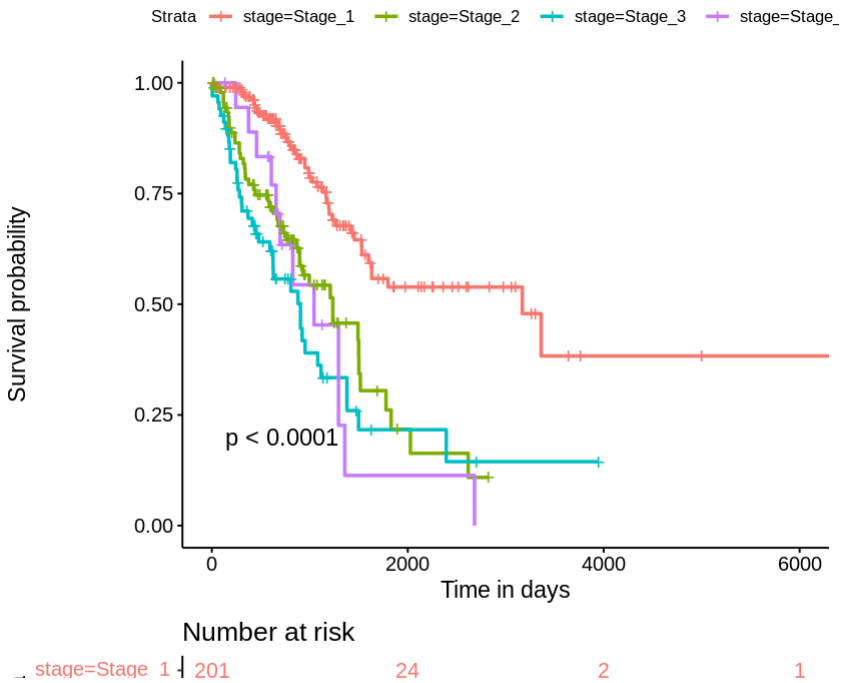


Develop survival model for tumor stage

```
#survival object
surv_object <- Surv(time = as.numeric(input_data_tr1$vital.time), event = input_data_tr1$OS)

# stage based
fit_s = survfit(surv_object ~ stage, data=input_data_tr1)

cox_s <-coxph(surv_object ~ stage, data=input_data_tr1)
```

Create kaplan-meier plot to visualize the startifcation of risk groups

```
# we produce a Kaplan Meier plot
ggsurvplot(fit_s, data=input_data_tr1, pval=TRUE, risk.table=TRUE, xlab="Time in days", risk.
```



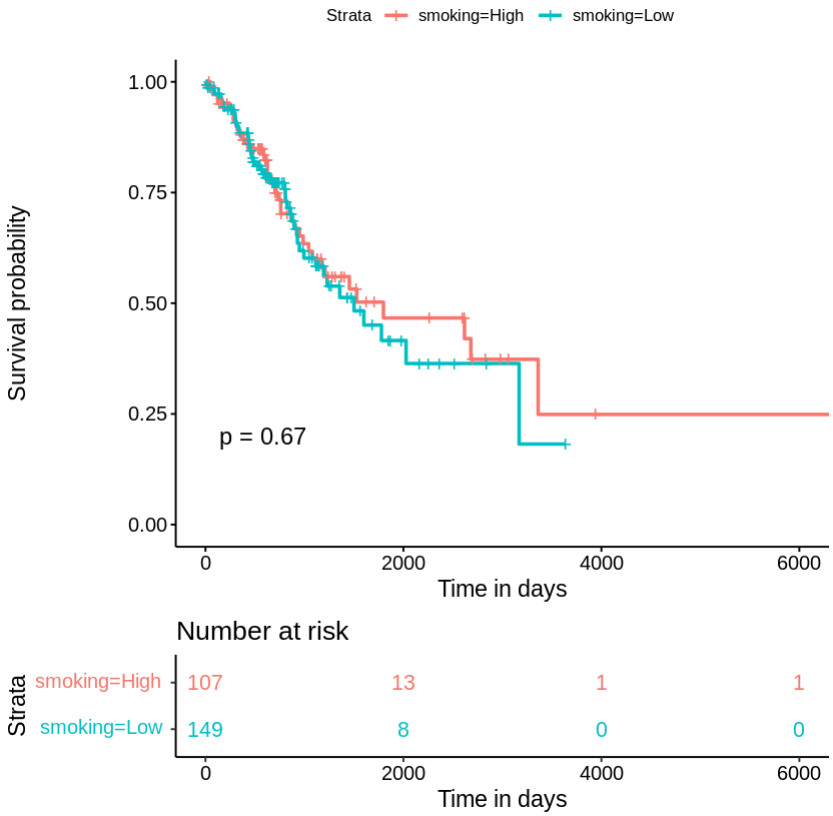
Devevlop model for smoking history

Time in days

```
#remove samples where cigarettes information is missing
input_data_tr2<-subset(input_data_tr1,cigarettes_per_day!="NA")

# Create age groups based of mean age (which is nearly 65)
smoking = as.data.frame(ifelse(input_data_tr2$cigarettes_per_day >= mean(as.numeric(input_dat
colnames(smoking) <- c("smoking")

#Add age group column to data
input_data_tr2 <- cbind(smoking,input_data_tr2)

head(input_data_tr2,5)
```

	smoking	id3	tissue_type	age_at_index	ethnicity	gender	race	vital_stat
	<chr>	<chr>	<chr>	<int>	<chr>	<chr>	<chr>	<cl
TCGA-05-4249-01A	High	TCGA-05-4249	Cancer	67	NA	male	NA	A
TCGA-05-	High	TCGA-05-	Cancer	68	NA	male	NA	A

```
#Create survival object
surv_object_sm <- Surv(time = as.numeric(input_data_tr2$vital.time), event = input_data_tr2$C

#fit model
fit_sm = survfit(surv_object_sm ~ smoking, data=input_data_tr2)

# Kaplan Meier plot
ggsurvplot(fit_sm, data=input_data_tr2, pval=TRUE, risk.table=TRUE, xlab="Time in days", risk
```



From the Univariate survival analysis based on few Clinical factors, it has been observed tumor stage is key prognostic factor for LUAD patients with p-value < 0.01.

Gender, age group and smoking (cigeretes) factors are not significant prognostic factors.

Thus, next we need to find molecular signature as prognostic factors

Univariate Survival analysis for Molecular/transcriptomics features

Here, we will perform univariate survival analysis for selected transcriptomics features.

We will perform univariate survival analysis on each feature in the data employing `for` loop using COXPH model

Here, I am also applying (log rank test) p-value (< 0.05) filetr criteria to filter only significant genes, eventually we are writing results into a file

```
# create a file to store results
write.table(cbind("Gene","Beta","HR","P-value","GP1","GP2","Hr-Inv-lst","Concordance","Std_Er
            file="OS-final-HRmedian-quantright.csv",row.names=F,col.names=F,sep = ',');

#Here features to compute survival start from 17th column onwards, which is transcriptomics f

for(i in seq(from=17, to=length(input_data_tr1), by=1))
{
  surv_object2 <- Surv(time = input_data_tr1$vital.time, event = input_data_tr1$OS)

#survival analysis: fits cox ph model to find HR for median cut
  fit1 <- survfit(surv_object2 ~ (input_data_tr1[,i])>(median(input_data_tr1[1,i])), data=inp
  summary(fit1);
  fit1.coxph <- coxph(surv_object2 ~ (input_data_tr1[,i])>(median(input_data_tr1[1,i])), data
# summary(fit1.coxph);
  first <- coef(summary(fit1.coxph))

#check whether the pvalue is significant (< 0.05) or not
  if((first[5]<=0.05)&&(!is.na(first[5]))&&(!is.na(first[2])))
  {write.table(cbind(colnames(input_data_tr1[i]),first[1],first[2],first[5],fit1$n[1],fit1$n[
               file="OS-final-HRmedian-quantright.csv",row.names=F,col.names=F,sep = ',',appe
  }
}
```

```
Warning message in coxph.fit(X, Y, istrat, offset, init, control, weights = weights, :
“Loglik converged before variable  1 ; coefficient may be infinite. ”
Warning message in coxph.fit(X, Y, istrat, offset, init, control, weights = weights, :
“Loglik converged before variable  1 ; coefficient may be infinite. ”
Warning message in coxph.fit(X, Y, istrat, offset, init, control, weights = weights, :
“Loglik converged before variable  1 ; coefficient may be infinite. ”
Warning message in coxph.fit(X, Y, istrat, offset, init, control, weights = weights, :
“Loglik converged before variable  1 ; coefficient may be infinite. ”
Warning message in coxph.fit(X, Y, istrat, offset, init, control, weights = weights, :
“Loglik converged before variable  1 ; coefficient may be infinite. ”
Warning message in coxph.fit(X, Y, istrat, offset, init, control, weights = weights, :
“Loglik converged before variable  1 ; coefficient may be infinite. ”
Warning message in coxph.fit(X, Y, istrat, offset, init, control, weights = weights, :
“Loglik converged before variable  1 ; coefficient may be infinite. ”
Warning message in coxph.fit(X, Y, istrat, offset, init, control, weights = weights, :
“Loglik converged before variable  1 ; coefficient may be infinite. ”
Warning message in coxph.fit(X, Y, istrat, offset, init, control, weights = weights, :
```

```
“Loglik converged before variable  1 ; coefficient may be infinite. ”
Warning message in coxph.fit(X, Y, istrat, offset, init, control, weights = weights, :
“Loglik converged before variable  1 ; coefficient may be infinite. ”
Warning message in coxph.fit(X, Y, istrat, offset, init, control, weights = weights, :
“Loglik converged before variable  1 ; coefficient may be infinite. ”
```

Now, let's see the univariate survival analysis results based on molecular/transcriptomics features.

```
Surv_res <- read.table("OS-final-HRmedian-quantright.csv", sep=",", header=TRUE, row.names=1)
dim(Surv_res)
head(Surv_res)
```

56 · 8

A data.frame: 6 × 8

	Beta	HR	P.value	GP1	GP2	Hr.Inv.lst	Concord
	<dbl>	<dbl>	<dbl>	<int>	<int>	<dbl>	<(
ENSG00000273797.1	0.6230904	1.864682	0.0025351948	328	54	0.5362845	0.556
ENSG00000269806.1	0.5763657	1.779559	0.0007110813	199	183	0.5619369	0.569
ENSG00000253953.2	0.3787163	1.460409	0.0399832945	289	93	0.6847399	0.530
ENSG00000223653.4	0.4087640	1.504956	0.0225879892	140	242	0.6644710	0.550
ENSG00000259915.2	0.4741341	1.606622	0.0375894113	336	46	0.6224238	0.517
ENSG00000261226.1	0.4536065	1.573979	0.0274864015	115	267	0.6353327	0.537

Sort features based on concordance index

```
# Sort features based on Corcordance
CI_features <- as.data.frame(Surv_res[order(Surv_res[,"Concordance"], decreasing =  TRUE),])
dim(CI_features)
head(CI_features)
```

56 · 8

A data.frame: 6 × 8

56 genes found to be significant (p-value < 0.05) prognostic factors based on univariate survival analysis. Top 6 genes based on the Concordance index reprsented in the above table.

Analyze Prognostic factors

From the univariate analysis, 56 genes, and tumor stage found to be significant prognostic factor. Next, it would be intersting to understand whether are good or bad prognostic factor. We can also apply more stringent criterias (HR (**Hazard ratio, Concordance Index**) to further filter or reduce signatures.

Here, I am applying HR criteria to reduce and understand the impact on prognosis. I will assign gene as bad prognostic factor (HR >1.2) and good prognostic factor Inverse_of_HR >1.2

```
#Bad Prognostic features based on HR >1.2
sig_surv_features <- CI_features[CI_features$Concordance >= 0.55, ]
dim(sig_surv_features)

survival_unfavorable <- sig_surv_features[sig_surv_features$HR >= 1.2, ]

dim(survival_unfavorable)
head(survival_unfavorable)
```

23 · 8

23 · 8

A data.frame: 6 × 8

	Beta	HR	P.value	GP1	GP2	Hr.Inv.lst	Concordanc
	<dbl>	<dbl>	<dbl>	<int>	<int>	<dbl>	<dbl
ENSG00000270806.1	0.5036575	1.654763	2.954669e-03	195	187	0.6043163	0.586916
ENSG00000226533.1	0.8045472	2.235684	1.111443e-04	129	253	0.4472904	0.586211
ENSG00000256069.6	0.6824047	1.978630	4.900799e-05	234	148	0.5054002	0.579325
ENSG00000250878.3	0.5208832	1.683514	1.971607e-03	254	128	0.5939957	0.577552

```
#Good Prognostic features based on inverse of HR >1.2
survival_favorable <- sig_surv_features[sig_surv_features$Hr.Inv.lst >= 1.2, ]
dim(survival_favorable)
head(survival_favorable)
```

0 · 8

A data.frame: 0 × 8

Beta	HR	P.value	GP1	GP2	Hr.Inv.lst	Concordance	Std_Error
<dbl>	<dbl>	<dbl>	<int>	<int>	<dbl>	<dbl>	<dbl>

After applying filter criteria, we found 23 genes as significant bad prognostic factors and none of genes able to pass filter criteria for good prognostic factor

Gene Annotation

After identifying significant prognostic genes. It is also important we must be able annotate to undrstand what are these transcripts . Here, I am using `org.Hs.eg.db` package to annotate our genes. We can also perform annotatation using other databases as well.

```
#install package
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("org.Hs.eg.db")
```

```
     Installing package into ‘/usr/local/lib/R/site-library’
     (as ‘lib’ is unspecified)

     'getOption("repos")' replaces Bioconductor standard repositories, see
     '?repositories' for details

     replacement repositories:
         CRAN: https://cran.rstudio.com


     Bioconductor version 3.13 (BiocManager 1.30.16), R 4.1.1 (2021-08-10)

     Installing package(s) 'BiocVersion', 'org.Hs.eg.db'

     also installing the dependencies ‘zlibbioc’, ‘GenomeInfoDbData’, ‘XVector’, ‘GenomeInfo


     Old packages: 'gert', 'openssl', 'rmarkdown', 'vroom', 'waldo', 'nlme'
```

```
library(org.Hs.eg.db)
keytypes(org.Hs.eg.db)
```

```
Loading required package: AnnotationDbi

Loading required package: stats4

Loading required package: BiocGenerics

Loading required package: parallel


Attaching package: ‘BiocGenerics’


The following objects are masked from ‘package:parallel’:

    clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
    clusterExport, clusterMap, parApply, parCapply, parLapply,
    parLapplyLB, parRapply, parSapply, parSapplyLB


The following objects are masked from ‘package:stats’:

    IQR, mad, sd, var, xtabs


The following objects are masked from ‘package:base’:

    anyDuplicated, append, as.data.frame, basename, cbind, colnames,
    dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
    grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
    order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
    rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
    union, unique, unsplit, which.max, which.min


Loading required package: Biobase

Welcome to Bioconductor

    Vignettes contain introductory material; view with
    'browseVignettes()'. To cite Bioconductor, see
    'citation("Biobase")', and for packages 'citation("pkgname")'.


Loading required package: IRanges

Loading required package: S4Vectors


Attaching package: ‘S4Vectors’


The following objects are masked from ‘package:base’:

    expand.grid, I, unname
```

```
#check names of columns that we have in database
```

```
columns(org.Hs.eg.db)
```

'ACCNUM' · 'ALIAS' · 'ENSEMBL' · 'ENSEMBLPROT' · 'ENSEMBLTRANS' · 'ENTREZID' · 'ENZYME' · 'EVIDENCE' · 'EVIDENCEALL' · 'GENENAME' · 'GENETYPE' · 'GO' · 'GOALL' · 'IPI' · 'MAP' · 'OMIM' · 'ONTOLOGY' · 'ONTOLOGYALL' · 'PATH' · 'PFAM' · 'PMID' · 'PROSITE' · 'REFSEQ' · 'SYMBOL' · 'UCSCKG' · 'UNIPROT'

```
#column we want to extract
#cols <- c("SYMBOL", "ENTREZID", "GENETYPE", "GOALL", "ONTOLOGY", "PATH")

cols <- c("SYMBOL", "ENTREZID", "GENETYPE", "GENENAME")


#get all important features as characters
CI_f<- as.character(row.names(CI_features))

# convert ensemble transcript IDs to ensemble IDs (since, above database have information reg
CI_f1 <- gsub("\\.[0-9]*$", "", CI_f)



# Perform annotation
annotated_genes <- as.data.frame(select(org.Hs.eg.db, keys=CI_f1, columns=cols, keytype="ENSE
head(annotated_genes, 20)
```

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

A data.frame: 20 × 5

	ENSEMBL	SYMBOL	ENTREZID	GENETYPE	GENENAME
	<chr>	<chr>	<chr>	<chr>	<chr>
1	ENSG00000270806	C17orf50	146853	protein-coding	chromosome 17 open reading frame 50
2	ENSG00000226533	BTBD9-AS1	101929425	ncRNA	BTBD9 antisense RNA 1
3	[illegible]	[illegible]	[illegible]	[illegible]	alpha-2-macroglobulin
4	ENSG00000250878	METTL21EP	121952	pseudo	methyltransferase like 21E,

Draw KM plot for a specific gene

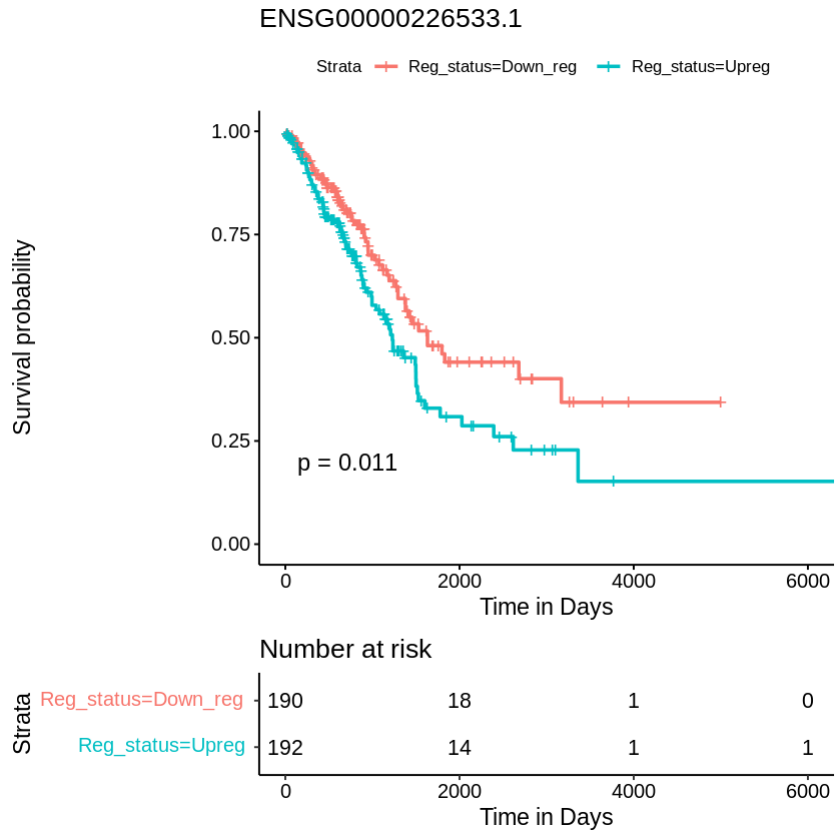
```
#Extract gene expression data for that gene
gene1 <- input_data_tr1$ENSG00000226533.1

#extract survival  information
os <- input_data_tr1$OS
os.time <- input_data_tr1$vital.time

# create a new column regulation status for the gene based on the cut of median value
reg_status = as.data.frame(ifelse(input_data_tr1$ENSG00000226533.1 >= median(as.numeric(input

# add column name
colnames(reg_status) <- c("Reg_status")

#combine information together to create a new data frame
gene1_data <- cbind(gene1,os, os.time, reg_status)

#print top rows
head(gene1_data)
```

A data.frame: 6 × 4

	gene1	os	os.time	Reg_status
	<chr>	<dbl>	<int>	<chr>
1	-0.138	0	1523	Down_reg
2	-0.269	0	607	Upreg
3	-0.251	0	1369	Upreg
4	-0.087	0	1126	Down_reg
5	-0.267	0	912	Upreg
6	-0.312	0	791	Upreg

```
#create survival object
surv_object2 <- Surv(time = gene1_data$os.time, event = gene1_data$os)
```

```
#fit model
fit_G <- survfit(surv_object2 ~ Reg_status , data=gene1_data)

#Draw KM plot
ggsurvplot(fit_G, data=gene1_data, pval=TRUE, xlab="Time in Days", risk.table=TRUE, title="EN
```



Multivariate Prognostic models

From the univariate analysis, we are able to identify 23 genes and stage as significant prognostic factors.

To understand whether these genes (top 6) and clinical factors also act as independent prognostic factor. We need to develop multivariate survival analysis.

Manipulate genes Ids (replace digits after decimals with none, since, code is not able to take it's original name as numeric factor)

```
nm <- names(input_data_tr1)
nm2 <- gsub("\\.[0-9]*$", "", nm)

# Recreate data with substituted column names
ip <- input_data_tr1
colnames(ip) <- nm2
```

```
#Print top rows
head(ip [16:20],5)
```

A data.frame: 5 × 5

	stage_n	ENSG00000259883	ENSG00000231981	ENSG00000263089	ENSG00000233540
	<dbl>	<chr>	<chr>	<chr>	<chr>
TCGA-05-4249-01A	1	-0.068	-0.121	-0.085	1.450
TCGA-05-4382-01A	1	-0.353	-0.157	-0.468	0.129
TCGA-05-4389-01A	1	-0.362	-0.250	-0.491	-0.040
TCGA-05-	1	-0.267	-0.250	-0.472	-0.455

```
head(survival_unfavorable)
```

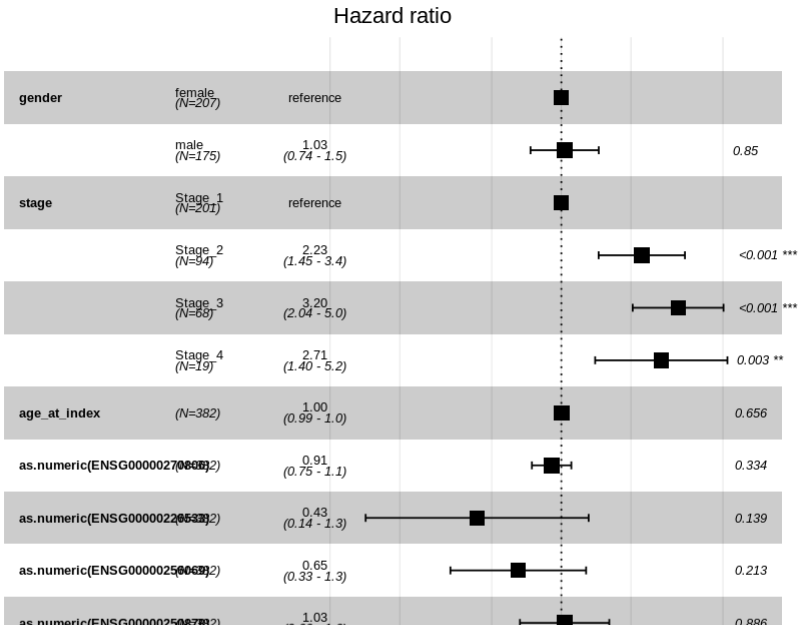
A data.frame: 6 × 8

	Beta	HR	P.value	GP1	GP2	Hr.Inv.lst	Concordanc
	<dbl>	<dbl>	<dbl>	<int>	<int>	<dbl>	<dbl
ENSG00000270806.1	0.5036575	1.654763	2.954669e-03	195	187	0.6043163	0.586916
ENSG00000226533.1	0.8045472	2.235684	1.111443e-04	129	253	0.4472904	0.586211
ENSG00000256069.6	0.6824047	1.978630	4.900799e-05	234	148	0.5054002	0.579325
ENSG00000250878.3	0.5208832	1.683514	1.971607e-03	254	128	0.5939957	0.577552

Multivariate model

First develop multivariate model with clinical factor and top molecular features (based on CI value)

```
# Fit a Cox proportional hazards model
surv_object_n <- Surv(time = ip$vital.time, event = ip$OS)
fit.coxph <- coxph(surv_object_n ~   gender + stage + age_at_index + as.numeric(ENSG000002708
ggforest(fit.coxph, data = ip)
```



From the multivariate analysis, we observed tumor stage is only significant prognostic factor, while none of the top 6 genes independently acts as significant prognostic factor.

Thus, we need to design better prognostic molecular signature. One approach is that we can develop PI (prognostic Index model), where we can combine multiple genes (considering their Coefficients values in univariate analysis)

`Coeff of gene1 * expression of Gene1 + Coeff of gene2 * expression of Gene2 + ... + Coeff of geneN * expression of GeneN`

Here, I am using same top six genes to build PI model

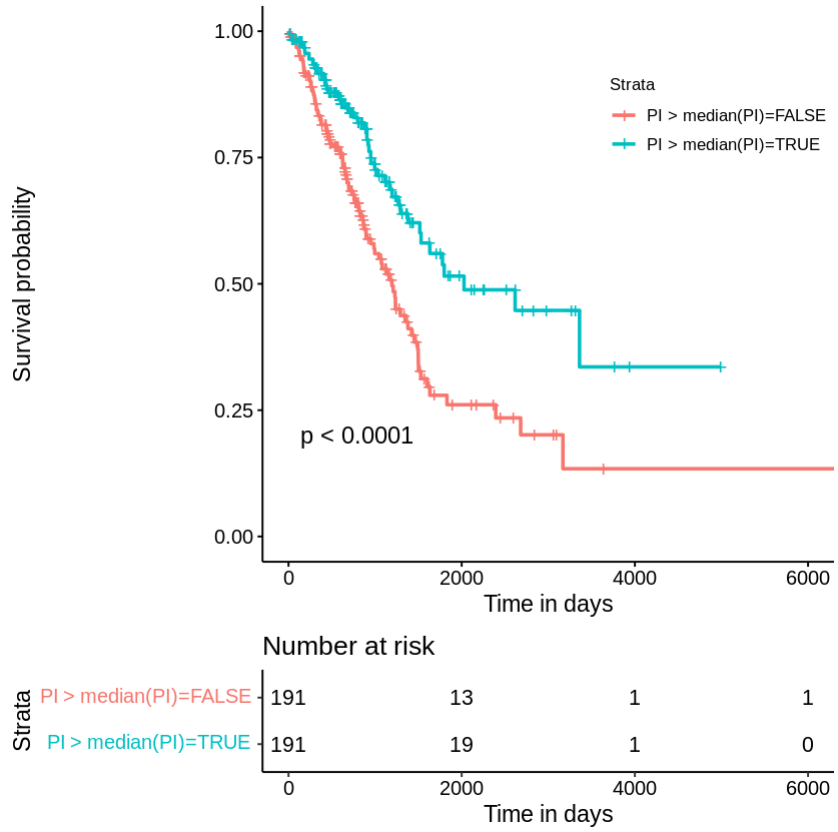
```
head(survival_unfavorable)
```

A data.frame: 6 × 8

	Beta	HR	P.value	GP1	GP2	Hr.Inv.lst	Concordanc
	<dbl>	<dbl>	<dbl>	<int>	<int>	<dbl>	<dbl
ENSG00000270806.1	0.5036575	1.654763	2.954669e-03	195	187	0.6043163	0.586916
ENSG00000226533.1	0.8045472	2.235684	1.111443e-04	129	253	0.4472904	0.586211
ENSG00000256069.6	0.6824047	1.978630	4.900799e-05	234	148	0.5054002	0.579325
ENSG00000250878.3	0.5208832	1.683514	1.971607e-03	254	128	0.5939957	0.577552

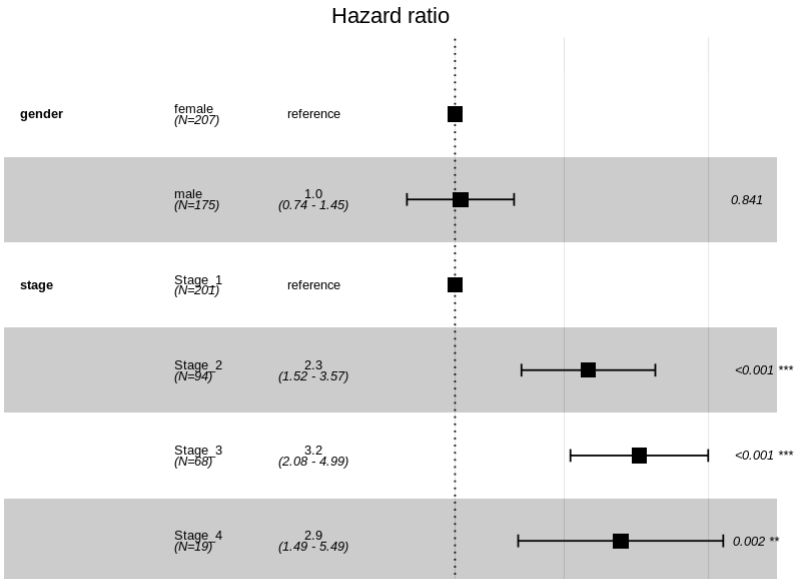
Prognostic Index (PI) model with top 6 genes

```
PI=0

PI= PI + as.numeric(ip$ENSG00000270806)*0.504 + as.numeric(ip$ENSG00000226533)*0.805 + as.num

ip$PI<-PI

surv_object_p <- Surv(time = ip$vital.time, event = ip$OS)

fit <- survfit(surv_object_p~(ip$PI>median(ip$PI)), data=ip)
ggsurvplot(fit, pval = TRUE, censor = TRUE, conf.int = F, xlab = "Time in days",  legend=c(0.
```



Multivariate model with PI model

```
# Fit a Cox proportional hazards model
surv_object_n <- Surv(time = ip$vital.time, event = ip$OS)
fit.coxph <- coxph(surv_object_n ~   gender + stage + age_at_index + ip$PI, data = ip )
ggforest(fit.coxph, data = ip)
```



PI model acts as significant prognostic factor both as univariate and multivariate models

Note: we can develop PI models with different combinations of features to obtain best model for publication purpose

Draw Boxplot for top genes

```
g1 <- as.data.frame(as.numeric(ip$ENSG00000270806))
g2 <- as.data.frame(as.numeric(ip$ENSG00000226533))
g3 <- as.data.frame(as.numeric(ip$ENSG00000256069))
g4 <- as.data.frame(as.numeric(ip$ENSG00000250878))
g5 <- as.data.frame(as.numeric(ip$ENSG00000224727))
g6 <- as.data.frame(as.numeric(ip$ENSG00000269806))
```

```
# create a new column risk status based on PI median value
Risk_status <- as.data.frame(ifelse(ip$PI >= median(as.numeric(ip$PI)), "High_Risk", "Low_Ris
```

```
#combine risk status with gene expression of key genes
new_data <- cbind(Risk_status, g1,g2,g3,g4,g5,g6)
```

```
# add column name
colnames(new_data) <- c("Risk", "ENSG00000270806", "ENSG00000226533", "ENSG00000256069", "ENS
```

```
# view top rows
head(new_data,4)
```

A data.frame: 4 × 7

	Risk	ENSG00000270806	ENSG00000226533	ENSG00000256069	ENSG00000250878	ENSG00
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	
1	Low_Risk	0.360	0.428	0.307	0.415	

Draw boxplot for these 6 genes and visualize the expression pattern among high and low risk groups

```
#install required package to shape dataframe
install.packages("reshape")
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

```
library("reshape")
```

```
#melt dataframe
df_m<- melt(new_data)
```

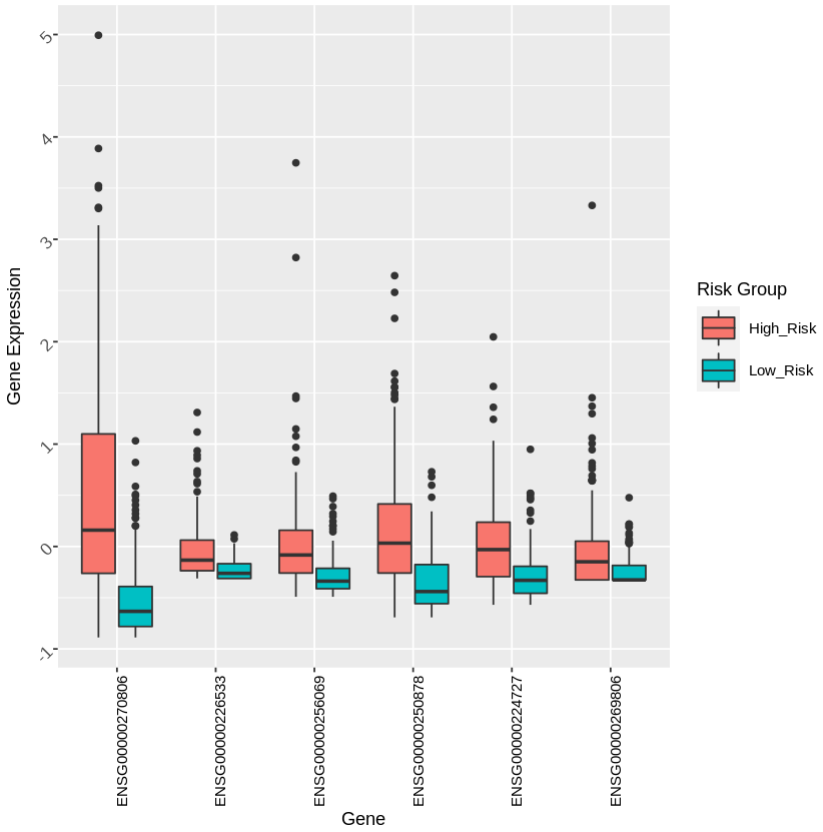
```
Using Risk as id variables
```

```
head(df_m,5)
```

A data.frame: 5 × 3

	Risk	variable	value
	<chr>	<fct>	<dbl>
1	Low_Risk	ENSG00000270806	-0.360
2	Low_Risk	ENSG00000270806	-0.428
3	Low_Risk	ENSG00000270806	-0.719
4	Low_Risk	ENSG00000270806	-0.888
5	Low_Risk	ENSG00000270806	-0.723

```
#create boxplot
b <- ggplot(df_m, aes(factor(variable), value)) + geom_boxplot(aes(fill = Risk))

#Add axis title, legend title and adjust font size
b + theme(legend.key.size = unit(0.9, "cm"), legend.text = element_text(color= "Black", size=
```



Model Evaluation on test data

For final validation, we must be validate our prognostic models on an independent dataset, which was not used during the feature selection and training or development of the model.

Here, I am using 20% data from the full data as test data

Prepare test data for survival analysis

```
#remove samples where OS.time is NA
input_data_te1<-subset(sel_te,vital.time!="NA")

#Check dimensions of data
dim(input_data_te1)
```

100 · 610

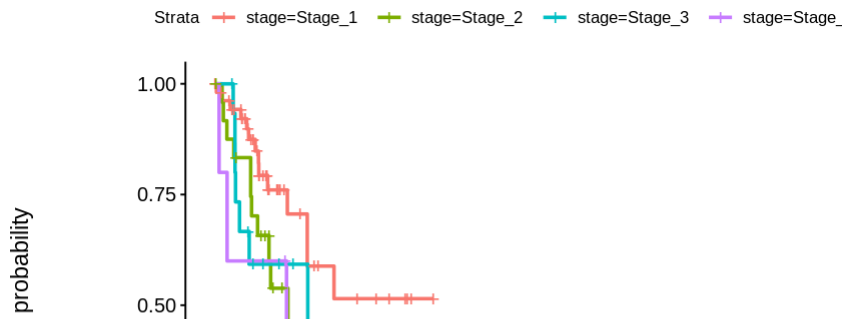
```
names1 <- names(input_data_te1)
names2 <- gsub("\\.[0-9]*$", "", names1)

te_ip <- input_data_te1
colnames(te_ip) <- names2

dim(te_ip)
head(te_ip, 5)
```

100 · 610

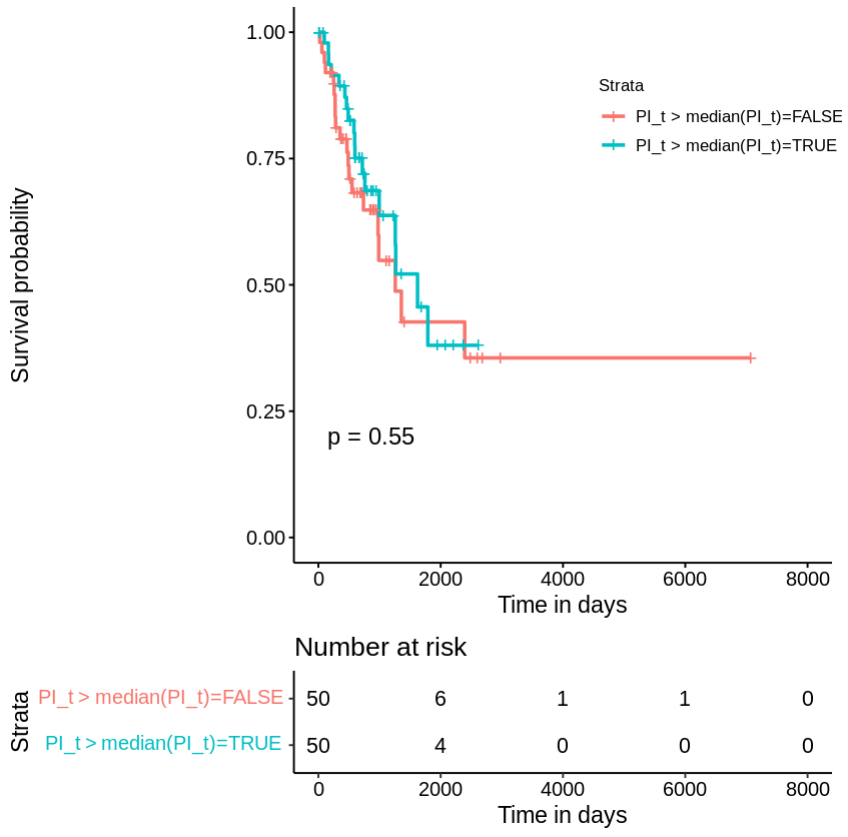
	id3	tissue_type	age_at_index	ethnicity	gender	race	vital_status	vita
	<chr>	<chr>	<int>	<chr>	<chr>	<chr>	<chr>	
TCGA-05-4417-01A	TCGA-05-4417	Cancer	51	NA	female	NA	Alive	
TCGA-05-4422-01A	TCGA-05-4422	Cancer	68	NA	male	NA	Alive	
TCGA-35-4122-01A	TCGA-35-4122	Cancer	69	not hispanic or latino	male	white	Alive	
TCGA-38-4625-01A	TCGA-38-4625	Cancer	66	not hispanic or latino	female	white	Alive	
TCGA-38-4631-01A	TCGA-38-4631	Cancer	72	not hispanic or latino	female	white	Dead	

```
# stage based
surv_object1 <- Surv(time = as.numeric(input_data_te1$vital.time), event = input_data_te1$OS)
fit_s1 = survfit(surv_object1 ~ stage, data=input_data_te1)

print(fit_s1)
# we produce a Kaplan Meier plot
ggsurvplot(fit_s1, data=input_data_te1, pval=TRUE, risk.table=TRUE, risk.table.col="stage")
```

```
Call: survfit(formula = surv_object1 ~ stage, data = input_data_te1)

                n events median 0.95LCL 0.95UCL
stage=Stage_1 54     14     NA    1258      NA
stage=Stage_2 25     11    995     582      NA
stage=Stage_3 16      9   1265     336      NA
stage=Stage_4  5      4    976     167      NA
```




```
#PI model
PI_t=0
PI_t= PI_t +  as.numeric(te_ip$ENSG00000270806)*0.504 + as.numeric(te_ip$ENSG00000226533)*0.8

te_ip$PI_t<-PI_t
surv_object_p_t <- Surv(time = te_ip$vital.time, event = te_ip$OS)

fit_t <- survfit(surv_object_p_t~(te_ip$PI_t>median(te_ip$PI_t)), data=te_ip)
ggsurvplot(fit_t, pval = TRUE, censor = TRUE, conf.int = F, xlab = "Time in days",  legend=c(
```

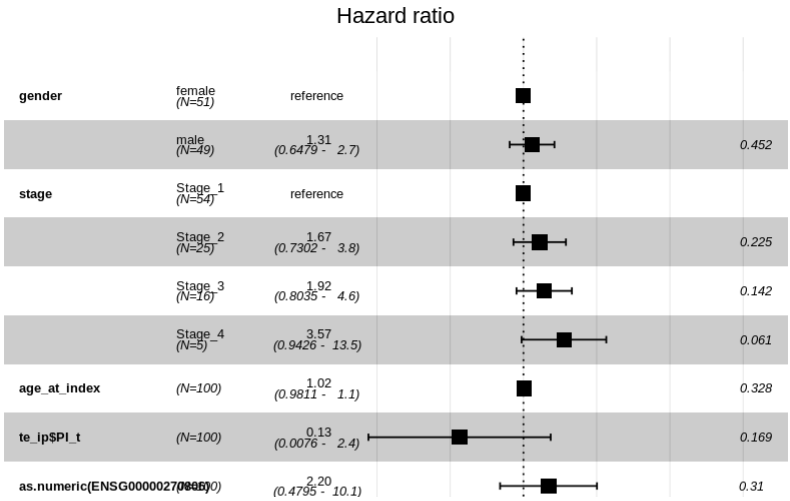



```
head(ip)
```

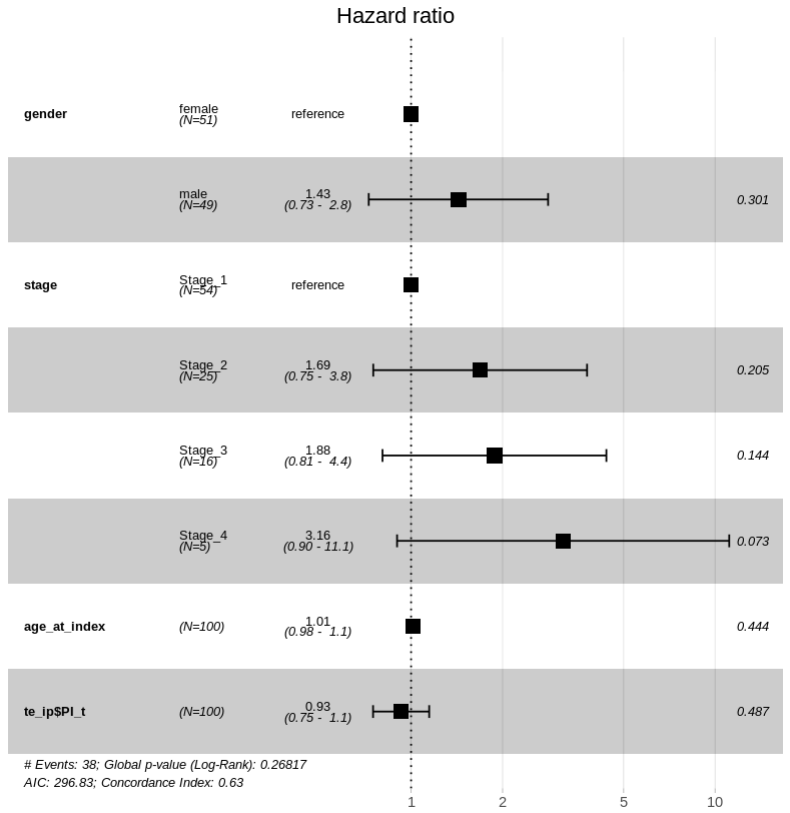
	id3	tissue_type	age_at_index	ethnicity	gender	race	vital_status	vita
	<chr>	<chr>	<int>	<chr>	<chr>	<chr>	<chr>	
TCGA-05-4249-01A	TCGA-05-4249	Cancer	67	NA	male	NA	Alive	
TCGA-05-4382-01A	TCGA-05-4382	Cancer	68	NA	male	NA	Alive	
TCGA-05-4389-01A	TCGA-05-4389	Cancer	70	NA	male	NA	Alive	
TCGA-05-4390-01A	TCGA-05-4390	Cancer	58	NA	female	NA	Alive	
TCGA-05-4420-01A	TCGA-05-4420	Cancer	41	NA	male	NA	Alive	
TCGA-05-4426-01A	TCGA-05-4426	Cancer	71	NA	male	NA	Alive	

Multivariate model

```
# Fit a Cox proportional hazards model
surv_object_p_t <- Surv(time = te_ip$vital.time, event = te_ip$OS)
fit.coxph_t <- coxph(surv_object_p_t ~   gender + stage + age_at_index + te_ip$PI_t + as.nume
ggforest(fit.coxph_t, data = te_ip)
```




```
# Multivariate Cox proportional hazards model
surv_object_p_t <- Surv(time = te_ip$vital.time, event = te_ip$OS)
fit.coxph_t <- coxph(surv_object_p_t ~   gender + stage + age_at_index + te_ip$PI_t, data = t
ggforest(fit.coxph_t, data = te_ip)
```

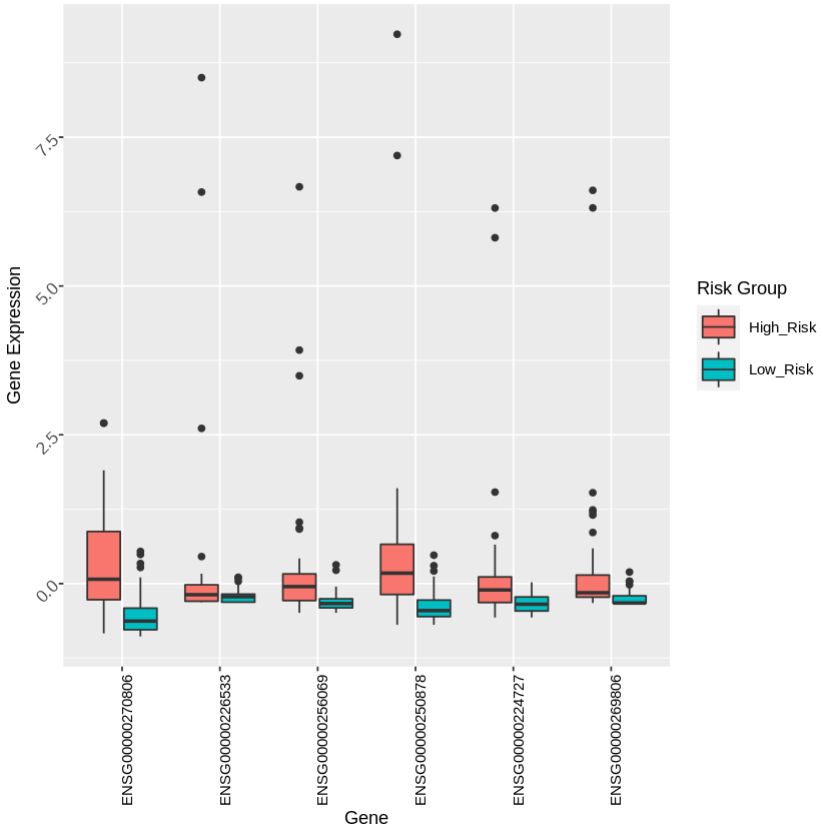


Boxplot for test data

```
g1t <- as.data.frame(as.numeric(te_ip$ENSG00000270806))
g2t <- as.data.frame(as.numeric(te_ip$ENSG00000226533))
g3t <- as.data.frame(as.numeric(te_ip$ENSG00000256069))
g4t <- as.data.frame(as.numeric(te_ip$ENSG00000250878))
g5t <- as.data.frame(as.numeric(te_ip$ENSG00000224727))
g6t <- as.data.frame(as.numeric(te_ip$ENSG00000269806))
```

```
# create a new column risk status based on PI median value
Risk_status_t <- as.data.frame(ifelse(te_ip$PI >= median(as.numeric(te_ip$PI)), "High_Risk",

#combine risk status with gene expression of key genes
new_data_t <- cbind(Risk_status_t, g1t,g2t,g3t,g4t,g5t,g6t)

# add column name
colnames(new_data_t) <- c("Risk", "ENSG00000270806", "ENSG00000226533", "ENSG00000256069", "E

#melt dataframe
df_m_t<- melt(new_data_t)

#create boxplot
bb <- ggplot(df_m_t, aes(factor(variable), value)) + geom_boxplot(aes(fill = Risk))

#Add axis title, legend title and adjust font size
bb + theme(legend.key.size = unit(0.9, "cm"), legend.text = element_text(color= "Black", size
```

```
Using Risk as id variables
```



Our model is not validated on test data based on this combination of features.

Note: For the final project work purpose, We can try more combination not just these 6 genes to validate model on test data

7. Classification Prediction models

Risk group classification Here, First Risk_status label will be assigned to the each samples based on the PI model. Thus "High_Risk =1" was assigned to samples PI > median(PI) value and Low_Risk=0 will be assigned to PI < median(PI) value. Next, for this excersie purpose, I am considering only 56 significant genes (based on univariate analysis as start input for Risk group classification. Since, survival analysis indicates stage is important prognostic factor. Thus, I have also considered tumor stage as one of input features.

Note: Here, I am creating binary classification models (High risk vs low risk)

Create training and test data for selected features

```
#create data frame for features only
top_CI <- as.data.frame(row.names(CI_features))

#add class column as well
#top1<-rbind(c("class"), top1)

#add column names
colnames(top_CI) <- c("ID")

#Write into a file
write.table(top_CI,'top_rf_features_list.txt', row.names=F,  sep='\t', quote=FALSE)
```

Prepare training and test data with selected features

```
# Prepare training data with only selected top features
input_data_tr1_t <- t(input_data_tr1)
input_data_te1_t <- t(input_data_te1)

#Prepare test data with selected features
tr_selected_data1 <- as.data.frame(input_data_tr1_t[row.names(input_data_tr1_t) %in% c(top_CI

#Prepare test data with selected features
te_selected_data1 <- as.data.frame(input_data_te1_t[row.names(input_data_te1_t) %in% c(top_CI

#Check dimensions of dataset
dim(tr_selected_data1)
dim(te_selected_data1)
```

56 · 382
56 · 100

```
#transpose train and test data
tr1 <-t(tr_selected_data1)
te1 <-t(te_selected_data1)

dim(tr1)
dim(te1)

tr_class <- as.data.frame(ip$stage_n)
te_class <- as.data.frame(te_ip$stage_n )

colnames(tr_class)<- c("T_stage")
colnames(te_class)<- c("T_stage")

dim(tr_class)
dim(te_class)

Risk_tr <- as.data.frame(new_data$Risk)
Risk_te <- as.data.frame(new_data_t$Risk)

colnames(Risk_tr)<- c("Risk")
colnames(Risk_te)<- c("Risk")

dim(Risk_tr)
dim(Risk_te)
```

$382 \cdot 56$
$100 \cdot 56$
$382 \cdot 1$
$100 \cdot 1$
$382 \cdot 1$
$100 \cdot 1$

```
# combine clinical data with gene expression for selected train & Test data
final_sel_tr_c <- cbind(Risk_tr, tr_class, tr1)
final_sel_te_c <- cbind(Risk_te, te_class, te1)

final_sel_tr_c$Risk_status<- recode(final_sel_tr_c$Risk,"c('High_Risk')=1;c('Low_Risk')=0")
final_sel_te_c$Risk_status <- recode(final_sel_te_c$Risk,"c('High_Risk')=1;c('Low_Risk')=0")


final_sel_tr_c1 <- final_sel_tr_c[2:ncol(final_sel_tr_c)]
final_sel_te_c1 <- final_sel_te_c[2:ncol(final_sel_te_c)]


#Check dimensions
dim(final_sel_tr_c1)
dim(final_sel_te_c1)


head(final_sel_tr_c1,3)
head(final_sel_te_c1,3)
```

```
#write into file
write.table(final_sel_tr_c1, file = "final_sel_tr_c1.txt", sep="\t", quote=F, row.names = F)
write.table(final_sel_te_c1, file = "final_sel_te_c1.txt", sep="\t", quote=F, row.names = F)
```

382 · 58
100 · 58

	T_stage	ENSG00000273797.1	ENSG00000269806.1	ENSG00000253953.2	ENSG000002236
	<dbl>	<chr>	<chr>	<chr>	<
TCGA-05-4249-01A	1	-0.353	-0.244	-0.652	-(
TCGA-05-4382-01A	1	-0.192	-0.326	3.319	-(
TCGA-05-4389-01A	1	-0.288	-0.243	-0.765	-(

	T_stage	ENSG00000273797.1	ENSG00000269806.1	ENSG00000253953.2	ENSG000002236
	<dbl>	<chr>	<chr>	<chr>	<
TCGA-05-4417-01A	1	-0.417	-0.326	0.596	-(
TCGA-05-4422-01A	1	-0.115	-0.326	1.108	-(
TCGA-35-4122-01A	1	-0.265	-0.326	-0.034	-(

RF model

Here, we will develop Random Forest model with `randomForest` package as well as `caret` package

```
install.packages("randomForest")
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

```
library("randomForest")
```

```
randomForest 4.6-14

Type rfNews() to see new features/changes/bug fixes.


Attaching package: ‘randomForest’


The following object is masked from ‘package:Biobase’:

    combine


The following object is masked from ‘package:BiocGenerics’:

    combine


The following object is masked from ‘package:ggplot2’:

    margin
```

Read training and test data for Classification

```
#Read training data
final_sel_tr_c1  <- read.table("final_sel_tr_c1.txt", sep = "\t", header=TRUE)
#Read training data
final_sel_te_c1  <- read.table("final_sel_te_c1.txt", sep = "\t", header=TRUE)
```

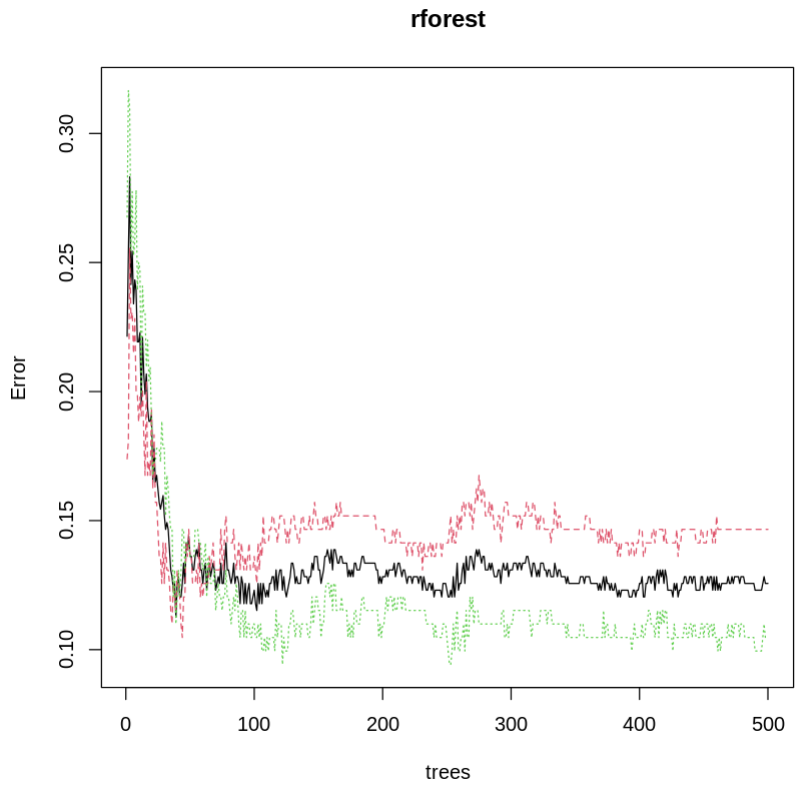
```
 number of trees
s <- as.numeric(500)
l <- as.matrix(new_tr)
 <- randomForest(as.factor(Risk_status) ~ ., data=final_sel_tr_c1, ntree=numtrees, metric="Ac
```

Model

```
rforest
```

```
      Call:
       randomForest(formula = as.factor(Risk_status) ~ ., data = final_sel_tr_c1,      ntree
                     Type of random forest: classification
                           Number of trees: 500
      No. of variables tried at each split: 7

              OOB estimate of  error rate: 10.73%
      Confusion matrix:
```

```
plot(rforest)
```

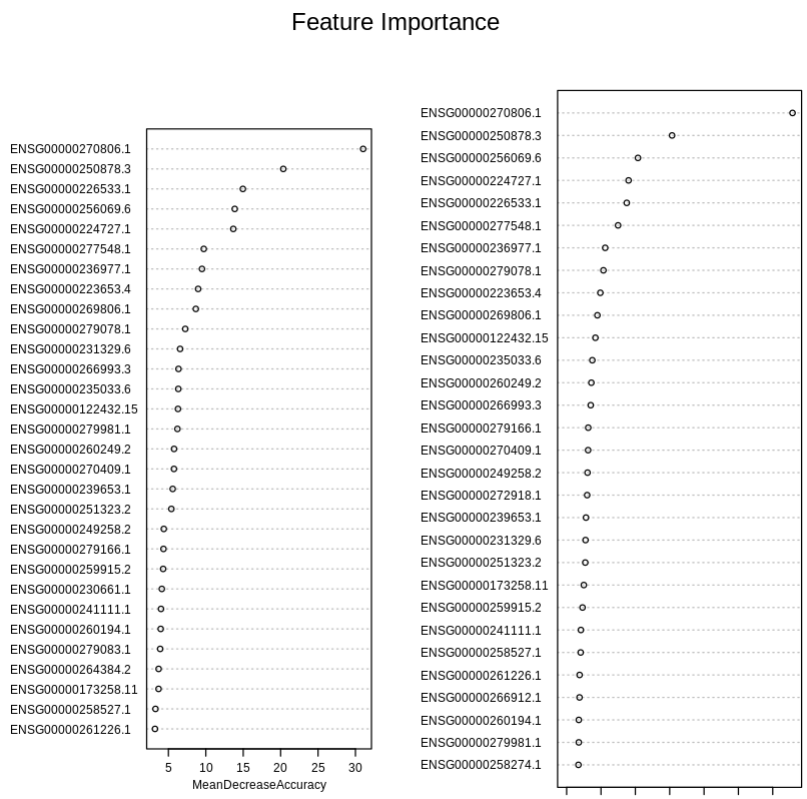


Create an output for features by feature importance

```
#Feature importance
varImpPlot(rforest, main='Feature Importance', pch=1, cex=0.6)

#Store features in a vaiable as features
rf_features <- varImpPlot(rforest, main='Feature Importance', pch=1, cex=0.6)

head(rf_features)

#Write features in a txt file format
write.table(rf_features,'RF_features_importance.txt', row.names=TRUE,col.names=NA, sep='\t',
```

A matrix: 6 × 2 of type dbl

	MeanDecreaseAccuracy	MeanDecreaseGini
T_stage	2.1874037	0.5049836
ENSG00000273797.1	2.0292900	1.6811999
ENSG00000269806.1	8.6651566	4.4797762
ENSG00000253953.2	0.6891555	1.6733874
ENSG00000223653.4	8.9750097	4.9038916
ENSG00000259915.2	4.2950597	2.3155350



Sort features based on the MeanDecrease Accuracy (for classification models) or %IncMSE (for regression models)

```
# Sort features based on meanDecreaseAccuracy value
sorted_features_rf <- as.data.frame(rf_features[order(rf_features[,"MeanDecreaseAccuracy"], d
#sorted_features_rf <- as.data.frame(rf_features[order(rf_features[,"%IncMSE"], decreasing =

head(sorted_features_rf)
```

A data.frame: 6 × 2

	MeanDecreaseAccuracy	MeanDecreaseGini
	<dbl>	<dbl>
ENSG00000270806.1	31.027577	32.884699
ENSG00000250878.3	20.355144	15.347057

Select top features based on MeandecreaseAccuracy score. Higher mean decrease accuracy score, more important feature would be classification model.

Note: Here, I am using filetr crietria >= 1.5. This filter criteria can be varied for selecting different subsets of features.

```
#select only top features (meanDecreaseaccuracy or %IncMSE >1.5)
top_features_rf <- sorted_features_rf[sorted_features_rf$MeanDecreaseAccuracy>1.2, ]
#top_features_rf <- sorted_features_rf[sorted_features_rf[1]>1.5, ]
dim(top_features_rf)
head(top_features_rf)
```

45 · 2

A data.frame: 6 × 2

	MeanDecreaseAccuracy	MeanDecreaseGini
	<dbl>	<dbl>
ENSG00000270806.1	31.027577	32.884699
ENSG00000250878.3	20.355144	15.347057
ENSG00000226533.1	14.953582	8.753131
ENSG00000256069.6	13.866291	10.383574
ENSG00000224727.1	13.672393	9.014240
ENSG00000277548.1	9.725275	7.486375

Install required packages

```
install.packages("e1071")
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependency ‘proxy’
```

```
library(e1071)
```

```
#Save the model
save(rforest, file = "RF.rda")


# Test set
test_set = final_sel_te_c1

#take data with features only
test_set1 <- (test_set[1:57])

head(test_set1)
```

	T_stage	ENSG00000273797.1	ENSG00000269806.1	ENSG00000253953.2	ENSG00000223653.4
	<int>	<dbl>	<dbl>	<dbl>	<dbl>
1	1	-0.417	-0.326	0.596	-0.256
2	1	-0.115	-0.326	1.108	-0.346
3	1	-0.265	-0.326	-0.034	-0.197
4	1	-0.331	-0.326	-0.721	-0.509
5	1	-0.417	-0.326	-0.240	-0.160
6	1	-0.173	-0.017	-0.624	-0.246

```
#Prediction on test data
predictions_RF <- predict(rforest, test_set1)


#Create·confusion·matrix
RF_cm_matrix·<-·confusionMatrix(as.factor(predictions_RF),·as.factor(test_set$Risk_status))
RF_cm_matrix
```

```
      Confusion Matrix and Statistics

                Reference
      Prediction  0  1
               0 46  8
               1  4 42

                     Accuracy : 0.88
                       95% CI : (0.7998, 0.9364)
          No Information Rate : 0.5
          P-Value [Acc > NIR] : 9.557e-16
```

```
#overall accuracy of the model on test data
acc <- as.data.frame(RF_cm_matrix$overall)
acc

#Results on test data
res <- as.data.frame(RF_cm_matrix$byClass)
res

#write test data into a file
write.table(res, file = "Results_test_set.txt", sep="\t", quote=F, row.names = F)
```

A data.frame: 7 × 1

	RF_cm_matrix$overall
	<dbl>
Accuracy	8.800000e-01
Kappa	7.600000e-01
AccuracyLower	7.997643e-01

RF with cross validation (3 fold)

```
#Create Cross validation (CV) object
train_control <- trainControl(method="repeatedcv", number=3)
```

```
#RF with CV
RF_cv <- train(as.factor(Risk_status) ~., method = "rf", trControl = train_control,  data=fir
```

```
RF_cv
```

```
Random Forest

382 samples
 57 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (3 fold, repeated 1 times)
Summary of sample sizes: 254, 256, 254
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.8246528  0.6493056
  29    0.9031085  0.8062169
  57    0.8979001  0.7958003

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 29.
```

```
#Save the model
save(RF_cv, file = "RF_cv.rda")
```

```
#Prediction·on·train·data
predictions_RF_cv_tr<-·predict(RF_cv,·final_sel_tr_c1[1:57])
```

```
#Create·confusion·matrix
RF_cm_matrix_cv_tr·<-·confusionMatrix(as.factor(predictions_RF_cv_tr),·as.factor(final_sel_tr
RF_cm_matrix_cv_tr
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 191   0
         1   0 191

               Accuracy : 1
                 95% CI : (0.9904, 1)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0
            Specificity : 1.0
         Pos Pred Value : 1.0
         Neg Pred Value : 1.0
             Prevalence : 0.5
         Detection Rate : 0.5
   Detection Prevalence : 0.5
      Balanced Accuracy : 1.0

       'Positive' Class : 0
```

```
#Prediction on test data
predictions_RF_cv_te <- predict(RF_cv, test_set1)
```

```
#Create confusion matrix of test data
RF_cm_matrix_cv_te <- confusionMatrix(as.factor(predictions_RF_cv_te), as.factor(test_set$Ris
RF_cm_matrix_cv_te
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 47 10
         1  3 40

               Accuracy : 0.87
```

Develop Logistic Regression model

Create Logistic Regression model with `glm` function on training data. With cv.glm we will perform cross validation on training data. Here, we are directly developing 3 fold cross validation prediction model

```
            Sensitivity : 0.9400
```

```
#Fit LR model
LR_model_cv <- train(factor(Risk_status) ~ ., data = final_sel_tr_c1, method = 'glm', metric
```

```
Warning message:
“glm.fit: algorithm did not converge”
Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”
Warning message:
“glm.fit: algorithm did not converge”
Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”
Warning message:
“glm.fit: algorithm did not converge”
Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”
Warning message:
“glm.fit: algorithm did not converge”
Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”
```

```
#Prediction on training data
predictions_LR_tr<- predict(LR_model_cv, final_sel_tr_c1[1:57])
```

```
#Confusion matrix
LR_cm_matrix_tr <- confusionMatrix(as.factor(predictions_LR_tr ), as.factor(final_sel_tr_c1$F
LR_cm_matrix_tr
```

```
      Confusion Matrix and Statistics

                Reference
      Prediction   0   1
               0 191   0
               1   0 191

                     Accuracy : 1
                       95% CI : (0.9904, 1)
          No Information Rate : 0.5
          P-Value [Acc > NIR] : < 2.2e-16

                      Kappa : 1

       Mcnemar's Test P-Value : NA

                  Sensitivity : 1.0
                  Specificity : 1.0
```

```
#Prediction on test data
predictions_LR<- predict(LR_model_cv, test_set1)
```

```
              Detection Rate : 0.5
```

```
#Confusion matrix
LR_cm_matrix <- confusionMatrix(as.factor(predictions_LR ), as.factor(test_set$Risk_status))
LR_cm_matrix
```

```
      Confusion Matrix and Statistics

                Reference
      Prediction  0  1
               0 48  8
               1  2 42

                     Accuracy : 0.9
                       95% CI : (0.8238, 0.951)
          No Information Rate : 0.5
          P-Value [Acc > NIR] : <2e-16

                      Kappa : 0.8

       Mcnemar's Test P-Value : 0.1138

                  Sensitivity : 0.9600
                  Specificity : 0.8400
               Pos Pred Value : 0.8571
               Neg Pred Value : 0.9545
                   Prevalence : 0.5000
               Detection Rate : 0.4800
         Detection Prevalence : 0.5600
            Balanced Accuracy : 0.9000

             'Positive' Class : 0
```

```
#overall accuracy of the model on test data
```

```
LR_acc <- as.data.frame(LR_cm_matrix$overall)
LR_acc

#Results on test data
LR_res <- as.data.frame(LR_cm_matrix$byClass)
LR_res
#write test data into a file
write.table(res, file = "LR_Results_test_set.txt", sep="\t", quote=F, row.names = F)
```

A data.frame: 7 × 1

	LR_cm_matrix$overall
	<dbl>
Accuracy	9.000000e-01
Kappa	8.000000e-01
AccuracyLower	8.237774e-01
AccuracyUpper	9.509953e-01
AccuracyNull	5.000000e-01
AccuracyPValue	1.531645e-17
McnemarPValue	1.138463e-01

A data.frame: 11 × 1

	LR_cm_matrix$byClass
	<dbl>
Sensitivity	0.9600000
Specificity	0.8400000
Pos Pred Value	0.8571429
Neg Pred Value	0.9545455
Precision	0.8571429
Recall	0.9600000
F1	0.9056604
Prevalence	0.5000000
Detection Rate	0.4800000
Detection Prevalence	0.5600000
Balanced Accuracy	0.9000000

SVM models

SVM radial & SVM linear Models

```
install.packages("kernlab")
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

```
library(kernlab)
```

```
Attaching package: ‘kernlab’


The following object is masked from ‘package:BiocGenerics’:

    type


The following object is masked from ‘package:ggplot2’:

    alpha
```

```
# Fit SVM RBF model
svm_RBF <- train(factor(Risk_status) ~ ., data = as.matrix(final_sel_tr_c1), method = 'svmRac
```

```
svm_RBF
```

```
Support Vector Machines with Radial Basis Function Kernel

382 samples
 57 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (3 fold, repeated 1 times)
Summary of sample sizes: 255, 255, 254
Resampling results across tuning parameters:

  C     Accuracy   Kappa
  0.25  0.7800197  0.5600432
  0.50  0.8350353  0.6700849
  1.00  0.8663673  0.7327502

Tuning parameter 'sigma' was held constant at a value of 0.01788908
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.01788908 and C = 1.
```

```
#Prediction on train data
predictions_SVC_RBF1 <- predict(svm_RBF, final_sel_tr_c1[1:57])
```

```
predictions_SVC_RBF1 <- predict(svm_RBF, final_set_tr_c1[1:9])
```

```
#Prediction on test data
predictions_SVC_RBF_t <- predict(svm_RBF, test_set1)
```

```
#Create confusion matrix
SVM_RF_cm_matrix <- confusionMatrix(as.factor(predictions_SVC_RBF_t), as.factor(test_set$Risk
SVM_RF_cm_matrix
```

```
     Confusion Matrix and Statistics

               Reference
     Prediction  0  1
              0 47 13
              1  3 37

                   Accuracy : 0.84
                     95% CI : (0.7532, 0.9057)
        No Information Rate : 0.5
        P-Value [Acc > NIR] : 1.303e-12

                      Kappa : 0.68

      Mcnemar's Test P-Value : 0.02445

                 Sensitivity : 0.9400
                 Specificity : 0.7400
              Pos Pred Value : 0.7833
              Neg Pred Value : 0.9250
                  Prevalence : 0.5000
              Detection Rate : 0.4700
        Detection Prevalence : 0.6000
           Balanced Accuracy : 0.8400

            'Positive' Class : 0
```

SVM -Linear model (SVM with Linear Kernal)

```
# Deevelop SVM linear model
svm_linear <- train(as.factor(Risk_status) ~ ., method = "svmLinear", trControl = train_contro
```

```
svm_linear
```

```
Support Vector Machines with Linear Kernel

382 samples
 57 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (3 fold, repeated 1 times)
```

```
#Prediction on train data
predictions_svm_l <- predict(svm_linear, final_sel_tr_c1[1:57])
```

```
  Accuracy   Kappa
```

```
#confusion matrix
SVM_Linear_cm_matrix_tr <- confusionMatrix(as.factor(predictions_svm_l), as.factor(final_sel_
SVM_Linear_cm_matrix_tr
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 190   0
         1   1 191

               Accuracy : 0.9974
                 95% CI : (0.9855, 0.9999)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9948

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9948
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9948
             Prevalence : 0.5000
         Detection Rate : 0.4974
   Detection Prevalence : 0.4974
      Balanced Accuracy : 0.9974

       'Positive' Class : 0
```

```
#Prediction on test data
predictions_svm_l_t <- predict(svm_linear, test_set1)
```

```
#confusion matrix
SVM_Linear_cm_matrix <- confusionMatrix(as.factor(predictions_svm_l_t  ), as.factor(test_set$
SVM_Linear_cm_matrix
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 50  7
         1  0 43

               Accuracy : 0.93
                 95% CI : (0.8611, 0.9714)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.86

 Mcnemar's Test P-Value : 0.02334

            Sensitivity : 1.0000
            Specificity : 0.8600
         Pos Pred Value : 0.8772
         Neg Pred Value : 1.0000
             Prevalence : 0.5000
         Detection Rate : 0.5000
   Detection Prevalence : 0.5700
      Balanced Accuracy : 0.9300
```

```
#7. Conclusion

**Prognostic Analysis**

First, I have identified 594 significant diffe
expressed genes between cancer & normal sample
selected genes were further used for downstrea
Based on univariate survival analysis of on TC
I have identified 56 significant (p-value <0.0
(transcriptomics) signatures. Subsequently, 23
bad prognostic signatures were filetered out (
p-value <0.05). Further Prognostic Index (PI)
developed using top 6 genes.
Besides, tumor stage observed to be the most s
prognostic signature.
Eventually,  multivariate survival analysis wa
understand independent prognostic potential of
clinical & molecular signatures.

Although, these models are not significantly p
validation data with this set of signatures. P
these models can be improved by applying diffe
combination of features for publication purpos

**Classification Prediction Model**

Next, Risk status labels (High Risk = 1 and Lo
```

7. Conclusion

Prognostic Analysis

First, I have identified 594 significant differentially expressed genes between cancer & normal samples. These selected genes were further used for downstream analyses. Based on univariate survival analysis of on TCGA-LUAD data, I have identified 56 significant (p-value <0.05) molecular (transcriptomics) signatures. Subsequently, 23 significant bad prognostic signatures were filetered out (HR >1.2 and p-value <0.05). Further Prognostic Index (PI) model was developed using top 6 genes. Besides, tumor stage observed to be the most significant prognostic signature. Eventually, multivariate survival analysis was performed to understand independent prognostic potential of these clinical & molecular signatures.

```
were assigned  to the samples in traing and te
genes based PI models.

To generate expression data for risk predictic
taken 56 genes (gene with p-value <0.05 based
analysis) and stage as input features to devel
models using Random Forest, Logistic regressic
SVMRadial and SVM linear. Most of these models
well on training data 96% accuarcy. These mode
reasonable performance on test data

What need to be done next to improve the perfo
groups prognostic prediction model for publica
project purpose:
1. Need more PI models (based on different com
features) which can also be validated on test
especially for the assignment of High-risk and
2. Take all genes as input features
3. Next, different feature selection methods n
implemented, e.g. Recursive feature selection
or RF, DGE analysis between High Risk & low ri
4. Since stage is observed to be key prognosti
Thus, we can also try to identify genes that c
early and late stage tumor samples.
5. Need validation on external data to make su
not overfitting on TCGA data
```

Although, these models are not significantly performed on validation data with this set of signatures. Performance of these models can be improved by applying different combination of features for publication purpose.

Classification Prediction Model

Next, Risk status labels (High Risk = 1 and Low Risk =0) were assigned to the samples in traing and test based on 6 genes based PI models.

To generate expression data for risk prediction, I have taken 56 genes (gene with p-value <0.05 based on univariate analysis) and stage as input features to develop training models using Random Forest, Logistic regression (LR), SVMRadial and SVM linear. Most of these models perform quite well on training data 96% accuarcy. These models also attain reasonable performance on test data

What need to be done next to improve the performance of Risk groups prognostic prediction model for publication or project purpose:

1. Need more PI models (based on different combination of features) which can also be validated on test dataset, especially for the assignment of High-risk and low risk group
2. Take all genes as input features
3. Next, different feature selection methods need to be implemented, e.g. Recursive feature selection (RFE) by SVM or RF, DGE analysis between High Risk & low risk groups
4. Since stage is observed to be key prognostic siganture. Thus, we can also try to identify genes that can classify early and late stage tumor samples.

5. Need validation on external data to make sure models are not overfitting on TCGA data

✓ 0s completed at 1:09 AM