SurvPredPipe

SurvPredPipe: a R-Package for the Computational Pipeline to Predict Survival Probability of Cancer Patients

Introduction:

The **SurvPredPipe** package is for a computational pipeline for the prediction Survival Probability of Cancer Patients. It performs various steps: Data Processing, Split data into training and test subset, Data Normalization, Select Significant features based on Univariate survival, Generate LASSO PI Score, Develop Prediction model for survival probability based on different features and draw survival curve based on predicted survival probability values and barplots for predicted mean and median survival time of patients.

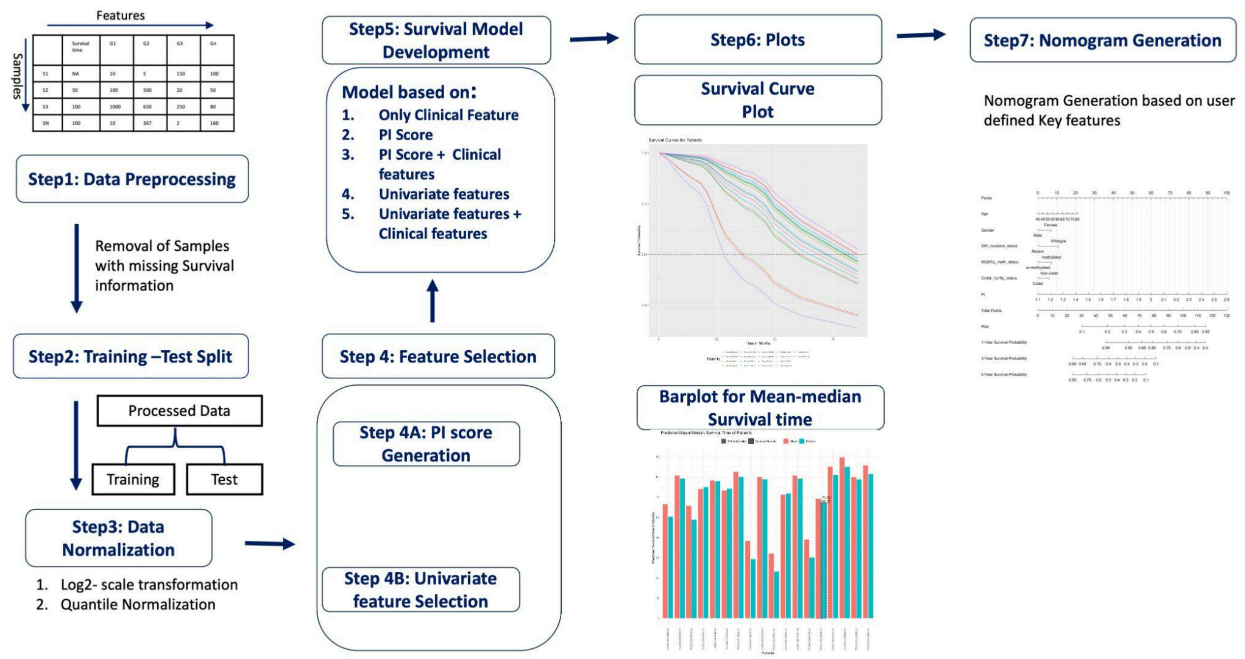

Features
Samples
Step1: Data Preprocessing
Removal of Samples with missing Survival information
Step2: Training –Test Split
Processed Data
Training
Test
Step3: Data Normalization
1. Log2- scale transformation
2. Quantile Normalization
Step5: Survival Model Development
Model based on:
1. Only Clinical Feature
2. PI Score
3. PI Score + Clinical features
4. Univariate features
5. Univariate features + Clinical features
Step 4: Feature Selection
Step 4A: PI score Generation
Step 4B: Univariate feature Selection
Step6: Plots
Survival Curve Plot
Barplot for Mean-median Survival time
Step7: Nomogram Generation
Nomogram Generation based on user defined Key features

Figure: The Workflow of SurvPredPipe representing different steps performed by various functions of SurvPredPipe package.

Key steps:

1. **Data Processing:** Removing samples where OS time information is missing and convert OS time in days into months
2. Train-Test Split: For Feature Selection and Model Development
3. Data Normalization: Log-scale transformation followed by Quantile Normalization
4. Univariate Feature Selection
5. Quantification of PI index per sample based on LASSO (glmnet method)

6. MTLR model Development based on:
    a. Clinical features
    b. PI Score
    c. PI index + Clin feature
    d. Univariate
    e. Univariate + Clin features
7. Survival Curve for Patients
8. BarPlots for Mean/Median Survival time
9. Nomogram

Final Results:

1. Table representing IBS score and C-Index for training model
2. Survival Probability curve plot per patient for test data
3. Mean and median Survival time value per patient
4. Table for Survival Probability
5. Nomogram to predict death risk, 1-year, 3-year and 5-year survival probability of patients based on user- selected features

Follow the Steps to Install package

~~~
``` #Step1: First Install remote package
install.packages("remotes")

#load remortes package
library("remotes")

#Step2: install SurvPredPipe package
remotes::install_github("hks5august/SurvPredPipe", local = TRUE)

# load package
library("SurvPredPipe")```
~~~

Workflow with an Example Data

#Load package

```
library(SurvPredPipe)
```

#Load other required packages

```
library(caret)
library(preprocessCore)
library(ggfortify)
library(survival)
library(survminer)
library(dplyr)
library(ggplot2)
library(ggfortify)
library(MASS)
library(MTLR)
library(dplyr)
library(SurvMetrics)
library(pec)
library(glmnet)
library(reshape2)
library(rms)
library(Matrix)
library(Hmisc)
library(survivalROC)
library(ROCR)
```

#set seed
```
set.seed(7)
```

#set path of the working directory where input data available
```
setwd()
```
Input Data
Input data: Example_TCGA_LGG_FPKM_data.txt

Example Input data: "Example_TCGA_LGG_FPKM_data.txt" is a tab separated file. It contains Samples (184 LGG Cancer Samples) in the rows and Features in the columns. Gene Expression is available in terms of FPKM values in the data.

- **Features information**: In the data there are 11 clinical + demographic, 4 types survival with time and event information and 19,978 protein coding genes.
- **Clinical and demographic features**: Clinical demographic features that are present in this example data include Age, subtype, gender, race, ajcc_pathologic_tumor_stage, histological_type, histological_grade, treatment_outcome_first_course, radiation_treatment_adjuvant, sample_type, type.
- **Types of Survival**: 4 types of Survival include OS (overall survival), PFS (progression-free survival), DSS (disease-specific survival), DFS (Disease-free survival). In the data, column names OS, PFS, DSS and DFS represent event information, while OS.time, PFS.time, DSS.time and DFS.time indicate survival time in days.

Let's Check input Data:

```
> #Check dimensions of input data
> dim(data)
[1]   184 19997

> #check top 3 rows and first 25 columns of the input data
> head(data[1:25],3)

                  Age subtype gender  race ajcc_pathologic_tumor_stage
histological_type histological_grade treatment_outcome_first_course
TCGA-TM-A7CA-01 44.94      PN   Male WHITE                          NA
Astrocytoma                 G2    Complete Remission/Response
TCGA-DU-A6S3-01 60.34      PN   Male WHITE                          NA
Oligodendroglioma                G2                Stable Disease
TCGA-CS-5390-01 47.80      PN Female WHITE                          NA
Oligodendroglioma                G2                          <NA>
                radiation_treatment_adjuvant sample_type type OS OS.time DSS
DSS.time DFI DFI.time PFI PFI.time   A1BG   A1CF     A2M  A2ML1 A3GALT2 A4GALT
TCGA-TM-A7CA-01                           NO     Primary  LGG  0    1058   0
1058   0     1058   0     1058 0.1166 0.0073 65.8414 0.6225  0.2736 0.4515
TCGA-DU-A6S3-01                           NO     Primary  LGG  0     656   0
656  NA       NA   0      656 0.0782 0.0070 42.7621 0.6587  0.1310 1.7230
TCGA-CS-5390-01                          YES     Primary  LGG  0      NA   0
NA   0       NA   0       NA 0.0877 0.0000 90.7590 0.7980  0.0653 1.0989
```

Step 1- Data Processing: This function converts OS time (in days) into months and then removes samples where OS/OS.time information is missing.
Here, we need to provide input data in tsv or txt format. Further, we needs to define col_num (column number at which clinical/demographic and survival information ends,e.g. 20, surv_time (name of column which contain survival time (in days) information, e.g. OS.time) and output file name, e.g. "New_data.txt"

#Data Processing:

```
SurvPredPipe::data_process_f(data="Example_TCGA_LGG_FPKM_data.txt",col_num=20,
surv_time="OS.time" , output="New_data.txt")
```

After data processing, `data_process_f` function will give us a new output file `"New_data.txt"`, which contains 176 samples. Thus, `data_process_f` function removes 8 samples where OS/OS time information is missing. Besides, here is a new 21st column in the data with column name "OS_month" where OS time is available in months.

Let's Check output of `data_process_f` function:

```
> #check output data
> output <- read.table("New_data.txt", sep="\t", header=TRUE, row.names=1,
check.names=FALSE)
> dim(output)
[1]   176 19998
> head(output[1:25],3)
                  Age subtype gender  race ajcc_pathologic_tumor_stage
histological_type histological_grade treatment_outcome_first_course
TCGA-TM-A7CA-01 44.94      PN   Male WHITE                          NA
Astrocytoma                G2    Complete Remission/Response
TCGA-DU-A6S3-01 60.34      PN   Male WHITE                          NA
Oligodendroglioma                G2                Stable Disease
TCGA-DU-8158-01 57.85    <NA> Female WHITE                          NA
Astrocytoma                G3                            <NA>
                radiation_treatment_adjuvant sample_type type OS OS.time DSS
DSS.time DFI DFI.time PFI PFI.time OS_month   A1BG   A1CF     A2M  A2ML1
A3GALT2
TCGA-TM-A7CA-01                           NO      Primary  LGG  0    1058   0
1058   0     1058   0     1058       35 0.1166 0.0073 65.8414 0.6225  0.2736
TCGA-DU-A6S3-01                           NO      Primary  LGG  0     656   0
656  NA       NA   0      656       22 0.0782 0.0070 42.7621 0.6587  0.1310
TCGA-DU-8158-01                         <NA>      Primary  LGG  1     155   1
155  NA       NA   1      155        5 0.2789 0.0052 91.1555 0.8302  0.0613
```

Step 2 - Split Data into Training and Test Subset: Before proceeding further, we need to split our data into training and test subset for the purpose of feature selection and model development. Here, we need output from the previous step as an input (which was "New_data.txt"). Next we need to define the fraction (e.g. 0.9) by which we want to split data into training and test. Thus, fraction=0.9 will split data into 90% training and 10% as test set. Besides, we also need to provide training and set output names (e.g. train_FPKM.txt,test_FPKM.txt)

# Split Data into Training and Test subset
```
SurvPredPipe::tr_test_f(data="New_data.txt",fraction=0.9,
train_data="train_FPKM.txt", test_data="test_FPKM.txt")
```

After the train-test split, we got two new outputs: "train_FPKM.txt", "test_FPKM.txt", where, train_FPKM.txt contains 158 samples and test_FPKM.txt contains 18 samples. Thus, tr_test_f function splits data into a 90:10 ratio.

Let's Check output of `tr_test_f` function:

```
> #load train_data output
> train_data <- read.table("train_FPKM.txt", sep="\t", header=TRUE,
row.names=1, check.names=FALSE)
> #check dimension of train_data output
> dim(train_data)
[1]   158 19998
> #load test_data output
> test_data <- read.table("test_FPKM.txt", sep="\t", header=TRUE, row.names=1,
check.names=FALSE)
#check dimension of train_data output
> dim(test_data)
[1]    18 19998
```

Step 3 - Data Normalization: Next to select features and develop ML models, data must be normalized. Since, expression is available in terms of FPKM values. Thus, `` `train_test_normalization_f` `` function will first convert FPKM value into log scale [log2(FPKM+1) followed by quantile normalization using the "`preprocessCore`" package. Here, training data will be used as a target matrix for quantile normalization. Here, we need to provide training and test datasets (that we obtained from the previous step of Train/Test Split). Further, we need to provide column number where clinical information ends (e.g. 21) in the input datasets. Besides, we also need to provide output files names (train_clin_data (which contains only Clinical information of training data), test_clin_data (which contains only Clinical information of training data), train_Normalized_data_clin_data (which contains Clinical information and normalized values of genes of training samples), test_Normalized_data_clin_data (which contains Clinical information and normalized values of genes of test samples).

# Data Normalization

```
``SurvPredPipe::train_test_normalization_f(train_data="train_FPKM.txt",test_dat
a="test_FPKM.txt", col_num=21, train_clin_data="Train_Clin.txt",
test_clin_data="Test_Clin.txt",
train_Normalized_data_clin_data="Train_Norm_data.txt",
test_Normalized_data_clin_data="Test_Norm_data.txt")``
```

After, running the function, we obtained 4 outputs:
1.Train_Clin.txt - Contains only Clinical features,
2. Test_Clin.txt- contains only Clinical features of Test samples;
3. Train_Norm_data.txt- Clinical features with normalized values of genes for training samples;
4. Test_Norm_data.txt - Clinical features with normalized values of genes for test samples.

Let's Check output of `train_test_normalization_f` function:

```
> # load normalized training data with clinical information
> train_Normalized_data_clin_data <- read.table("Train_Norm_data.txt",
sep="\t", header=TRUE, row.names=1, check.names=FALSE)

> # Check dimensions of normalized training data
> dim(train_Normalized_data_clin_data)
[1]   158 19998

> # View top 2 rows of training data with first 25 columns
> head(train_Normalized_data_clin_data[1:25],2)

                  Age subtype gender  race ajcc_pathologic_tumor_stage
histological_type histological_grade treatment_outcome_first_course
TCGA-CS-5396-01 53.11      PN Female WHITE                          NA
Oligodendroglioma                G3                             <NA>
TCGA-DU-A76L-01 54.27      ME   Male WHITE                          NA
Oligodendroglioma                G3            Progressive Disease
                radiation_treatment_adjuvant sample_type type OS OS.time DSS
DSS.time DFI DFI.time PFI PFI.time OS_month  A1BG A1CF     A2M A2ML1 A3GALT2
TCGA-CS-5396-01                          YES     Primary  LGG  0    1631   0
1631  NA       NA   0     1631       54 0.294    0  71.912 0.286   0.114
TCGA-DU-A76L-01                         <NA>     Primary  LGG  1     814   1
814  NA       NA   1      410       27 0.144    0 233.872 0.108   0.227

> #load normalized test data with clinical information
> test_Normalized_data_clin_data<- read.table("Test_Norm_data.txt", sep="\t",
header=TRUE, row.names=1, check.names=FALSE)

> # Check dimensions of normalized test data
> dim(test_Normalized_data_clin_data)
[1]    18 19998

> # View top 2 rows of test data with first 25 columns
> head(test_Normalized_data_clin_data[1:25],2)
                  Age subtype gender  race ajcc_pathologic_tumor_stage
histological_type histological_grade treatment_outcome_first_course
TCGA-DH-A669-02 70.66      PN   Male WHITE                          NA
Oligodendroglioma                G3                 Stable Disease
TCGA-WY-A859-01 34.58    <NA> Female WHITE                          NA
Astrocytoma                G2                 Stable Disease
                radiation_treatment_adjuvant sample_type type OS OS.time DSS
DSS.time DFI DFI.time PFI PFI.time OS_month  A1BG  A1CF    A2M A2ML1 A3GALT2
TCGA-DH-A669-02                         <NA>   Recurrent  LGG  1     919   0
919  NA       NA   1      260       30 0.301 0.009 60.121 0.401   0.000
TCGA-WY-A859-01                         <NA>     Primary  LGG  0    1213   0
1213  NA       NA   0     1213       40 0.081 0.000 31.696 1.748   0.095
```

Step 4a - Prognostic Index (PI) Score Calculation:
Next to create a survival model, we will create a Prognostic Index (PI) Score. PI score is calculated based on the expression of the features selected by the LASSO regression model and their beta coefficients. For instance, 5 features (G1, G2, G3, G4, and G5 and their coefficient values are B1, B2, B3, B4, and B5, respectively) selected by the LASSO method. Then PI score will be computed as following:

PI score = G1*B1 + G2*B2 + G3 * B3 + G4*B4+ G5*B5

Here, we need to provide Normalized training (Train_Norm_data.txt) and test data (Test_Norm_data.txt)as input data that we have obtained from the previous function "`train_test_normalization_f`". Further, we need to provide col_num n column number at which clinical features ends (e.g. 21), nfolds (number of folds e.g. 5) for the LASSO regression method to select features. We implemented LASSO using the "`glmnet`" package. Further, we need to provide surv_time (name of column containing survival time in months, e.g. OS_month) and surv_event (name of column containing survival event information, e.g. OS) information in the data. Besides, we also need to provide names and training and test output file names to store data containing LASSO genes and PI values.

# Feature Selection using LASSO Regression and Prognostic Index (PI) Score Calculation
```
SurvPredPipe::Lasso_PI_scores_f(train_data="Train_Norm_data.txt",test_data="Tes
t_Norm_data.txt", nfolds=5, col_num=21, surv_time="OS_month" , surv_event="OS"
, train_PI_data="Train_PI_data.txt", test_PI_data="Test_PI_data.txt" )
```

Thus, the `Lasso_PI_scores_f` gave us following outputs:

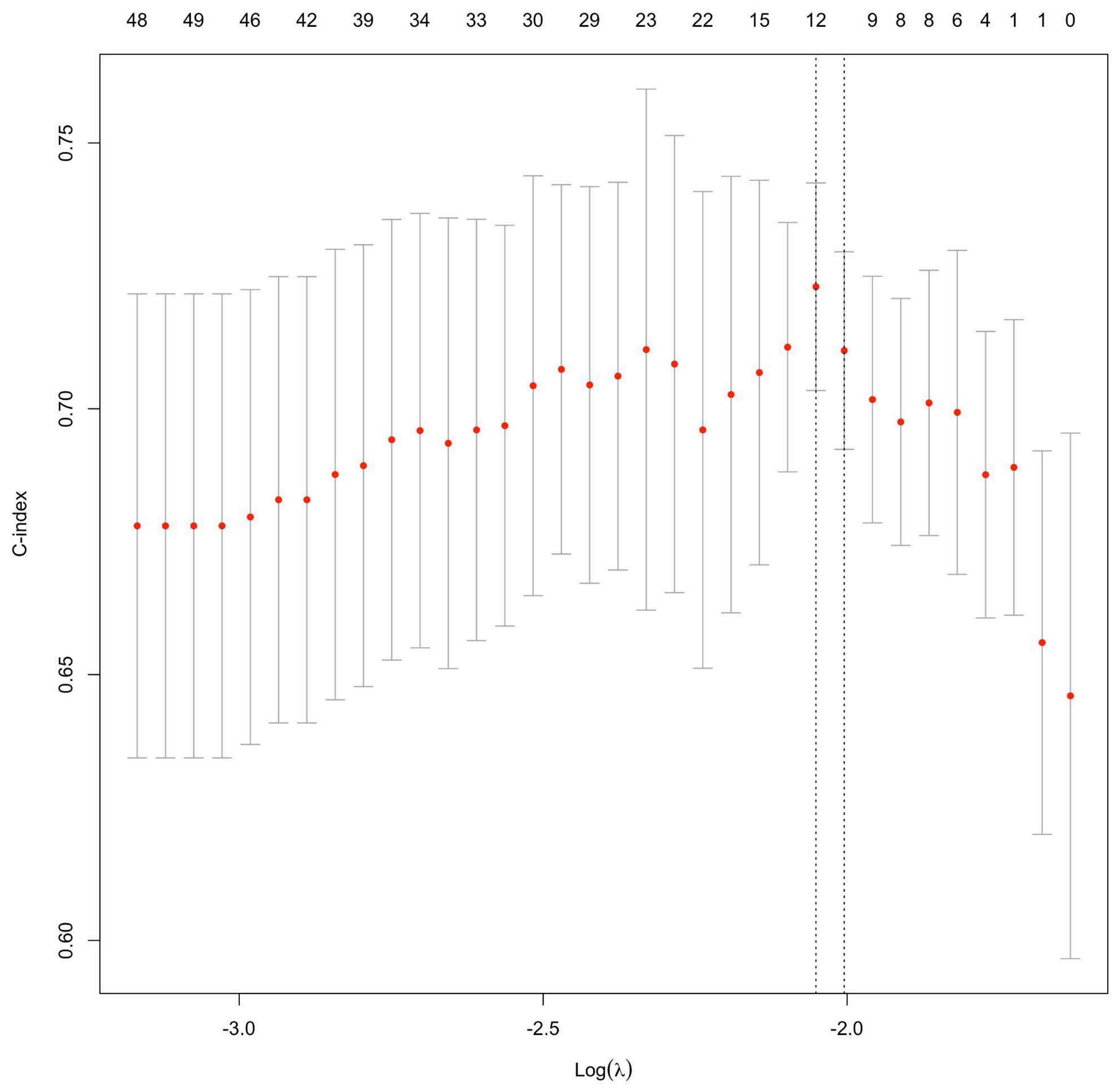
1. **Train_Lasso_key_variables.txt**: List of features selected by LASSO and their beta coefficient values

Let's Check output of `Lasso_PI_scores_f` function:

```
> # Load LASSO selected variables file
> Lasso_key_variables <- read.table("Train_Lasso_key_variables.txt", sep="\t",
header=TRUE, row.names=1, check.names=FALSE)

> # Check dimension
> dim(Lasso_key_variables)
[1] 12  1

> # view top of file
> head(Lasso_key_variables)
             coeff
```

```
ALG6        0.037
ARHGAP11A  0.096
DESI1      -0.028
GALNT7      0.101
GJD3        0.284
GPC1        0.005
```

2. **Train_Cox_Lasso_Regression_lamda_plot.jpeg**: Lasso Regression Lambda plot.



3. **Train_PI_data.txt**: It contains expression of genes selected by LASSO and PI score in the last column for training samples.

4. **Test_PI_data.tx**t: It contains expression of genes selected by LASSO and PI score in the last column for test samples.

```
> # Load train data containing LASSO selected genes and PI Value
> train_PI_data <- read.table("Train_PI_data.txt", sep="\t", header=TRUE,
row.names=1, check.names=FALSE)

> #check dimensions
> dim(train_PI_data)
[1] 158  15

> #View top 2 rows
> head(train_PI_data,2)
                OS OS_month  ALG6 ARHGAP11A  DESI1 GALNT7  GJD3   GPC1
H2BC5 HOXD12 RNF185 TANGO2    UNG ZNF648       PI

TCGA-CS-5396-01  0       54 4.705     1.241 24.866  5.760 0.074 20.434
4.004  0.019 19.033  1.660 11.262  0.115 0.287790

TCGA-DU-A76L-01  1       27 3.392     3.124  9.048  2.927 2.521 61.482
5.335  0.087  9.067  0.897 14.827  0.055 1.617081

> # Load test data containing LASSO selected genes and PI Value
> test_PI_data<- read.table("Test_PI_data.txt", sep="\t", header=TRUE,
row.names=1, check.names=FALSE)

> #check dimensions
> dim(test_PI_data)
[1] 18 15

> #View top 2 rows
> head(test_PI_data,2)

                OS OS_month  ALG6 ARHGAP11A  DESI1 GALNT7  GJD3   GPC1
H2BC5 HOXD12 RNF185 TANGO2    UNG ZNF648        PI

TCGA-DH-A669-02  1       30 2.919     1.288 18.533  2.323 0.295 54.726
8.651  0.136 11.046  1.780 20.197  0.089  0.486198

TCGA-WY-A859-01  0       40 1.433     0.279 23.926  1.066 0.117 39.250
4.240  0.006 13.179  1.829  9.860  0.003 -0.275303
```

Step 4b - Univariate Survival Significant Feature Selection:
Besides PI score, with the "`Univariate_sig_features_f`" function of SurvPredPipe package, we can select significant (p-value <0.05) features based on univariate survival analysis.

These features are selected based on their capability to stratify high-risk and low-risk survival groups using the cut off value of their median expression.
Here, we need to provide Normalized training (Train_Norm_data.txt) and test data (Test_Norm_data.txt)as input data that we have obtained from the previous function "`train_test_normalization_f`". Further, we need to provide a "`col_num`" (e.g 21) column number at which clinical features end. Further, we need to provide `surv_time` (name of column containing survival time in months, e.g. OS_month) and `surv_event` (name of column containing survival event information, e.g. OS) information in the data. Besides, we also need to provide names and training and test output file names to store data containing expression of selected genes.

# Feature selection using Univariate Survival Analysis

```
`SurvPredPipe::Univariate_sig_features_f(train_data="Train_Norm_data.txt",
test_data="Test_Norm_data.txt", col_num=21, surv_time="OS_month" ,
surv_event="OS" ,output_univariate_train="Train_Uni_sig_data.txt",
output_univariate_test="Test_Uni_sig_data.txt")`
```

Thus, the `Univariate_sig_features_f` function gave us following outputs:

1. **Univariate_Survival_Significant_genes_List.txt**: a table of univariate significant genes along with their corresponding coefficient values, HR value, P-values, C-Index values.

Let's Check output of `Lasso_PI_scores_f` function:

```
> #Load list of significant genes selected by Univariate Survival analysis
> Univariate_Survival_Significant_genes_List<-
read.table("Univariate_Survival_Significant_genes_List.txt", sep="\t",
header=TRUE, row.names=1, check.names=FALSE)

> # Check dimensions
> dim(Univariate_Survival_Significant_genes_List)
[1] 2391    8

> #View top 5 rows of Univariate Significant genes results
> head(Univariate_Survival_Significant_genes_List,5)
             Beta        HR     P-value GP1 GP2 Hr-Inv-lst Concordance
Std_Error

A2ML1   -0.9255326 0.3963203 0.016055099  20 138  2.5232117   0.5578119
0.03819547
AADACL2  1.3245266 3.7604047 0.001723308 148  10  0.2659288   0.5620198
0.03521353
AARS1   -0.7793207 0.4587175 0.016107789  74  84  2.1799910   0.6337358
0.04249702
```

```
ABCA12   1.3690240 3.9315117 0.004805718 150   8  0.2543551   0.5539700
0.03202036
ABCA6    0.9601084 2.6119796 0.038305760 136  22  0.3828514   0.5554336
0.03656689
```

2. **Train_Uni_sig_data.txt:** It contains expression of significant genes selected by univariate survival analysis for training samples.

```
> #Load training data with univariate significant genes
> train_Uni_sig_data <- read.table("Train_Uni_sig_data.txt", sep="\t",
header=TRUE, row.names=1, check.names=FALSE)

> # Check dimensions of data
> dim(train_Uni_sig_data )
[1]  158 2391

> # View top rows of training data
> head(train_Uni_sig_data[1:20],2)
                A2ML1 AADACL2  AARS1 ABCA12 ABCA6 ABCC11 ABCC3 ABCC9 ABHD10
ABHD11   ABI1 ABI3BP ABLIM1 AC004997.1 AC005832.4 AC006030.1 AC011455.2
AC012309.1
TCGA-CS-5396-01 0.286       0 33.197  0.214 0.403  0.157 1.152 0.630 12.924
6.461 22.471  0.541 14.608      0.000          0      0.006          0
0.015
TCGA-DU-A76L-01 0.108       0 26.215  0.298 0.231  0.059 5.263 0.371  9.812
7.890 14.744  0.877  9.228      0.005          0      0.010          0
0.007
                AC091057.5 AC093323.1
TCGA-CS-5396-01          0      5.222
TCGA-DU-A76L-01          0      8.101
```

3. **Test_Uni_sig_data.txt:** It contains expression of significant genes selected by univariate survival analysis for test samples.

```
> #Load test data with univariate significant genes
> test_Uni_sig_data<- read.table("Test_Uni_sig_data.txt", sep="\t",
header=TRUE, row.names=1, check.names=FALSE)

> # Check dimensions of data
> dim(test_Uni_sig_data)
[1]   18 2391

> # View top rows of test data
> head(test_Uni_sig_data[1:20],2)
```

```
                A2ML1 AADACL2  AARS1 ABCA12 ABCA6 ABCC11 ABCC3 ABCC9 ABHD10
ABHD11   ABI1 ABI3BP ABLIM1 AC004997.1 AC005832.4 AC006030.1 AC011455.2
AC012309.1
TCGA-DH-A669-02 0.401       0 37.720  0.078 0.011  0.073 0.338 0.194 14.795
17.986 20.666  0.272 13.782      0.000          0          0          0
0
TCGA-WY-A859-01 1.748       0 36.523  0.023 0.032  0.090 0.122 0.575 15.203
6.817 31.915  0.341 17.683      0.002          0          0          0
0
                AC091057.5 AC093323.1
TCGA-DH-A669-02          0      4.955
TCGA-WY-A859-01          0     10.609
```

Step 5 - Prediction model development for survival probability of patients
After selecting significant or key features using LASSO or Univariate survival analysis, next we want to develop an ML prediction model to predict survival probability of patients. `MTLR_pred_model_f` function of SurvPredPipe give us multiple options to develop models including Only Clinical features (Model_type=1), PI score (Model_type=2), PI Score + Clinical features (Model_type=3), Significant Univariate features (Model_type=4), Significant Univariate features Clinical features (Model_type=5) using MTLR package. Further, here, we were interested in developing a model based on PI score. Thus, we need to provide following inputs: (1) Training data with only clinical features, (2) Test data with only clinical features, (3) Model type (e.g. 2, since we want to develop model based on PI score), (4) Training data with PI score , (5) Test data with PI score, (6) Clin_Feature_List (e.g. Key_PI_list.txt), a list of features which will be used to build model . Furthermore, we also need to provide surv_time (name of column containing survival time in months, e.g. OS_month) and surv_event (name of column containing survival event information, e.g. OS) information in the clinical data

# Survival Model Development and Prediction of Survival, survival probability for Individual Patients using Selected Features employing MTLR
# Survival Model Development and Prediction of Survival, survival probability for Individual Patients using Selected Features employing MTLR

```
# Model for only Clinical features
SurvPredPipe::MTLR_pred_model_f(train_clin_data = "Train_Clin.txt",
test_clin_data = "Test_Clin.txt", Model_type = 1, train_features_data =
"Train_Clin.txt", test_features_data = "Test_Clin.txt" ,
Clin_Feature_List="Key_Clin_feature_list.txt", surv_time="OS_month",
surv_event="OS")



# Model for PI
SurvPredPipe::MTLR_pred_model_f(train_clin_data = "Train_Clin.txt",
test_clin_data = "Test_Clin.txt", Model_type = 2, train_features_data =
"Train_PI_data.txt", test_features_data = "Test_PI_data.txt" ,
Clin_Feature_List="Key_PI_list.txt", surv_time="OS_month", surv_event="OS")



# Model for Clinical features + PI
SurvPredPipe::MTLR_pred_model_f(train_clin_data = "Train_Clin.txt",
test_clin_data = "Test_Clin.txt", Model_type = 3, train_features_data =
"Train_PI_data.txt", test_features_data = "Test_PI_data.txt" ,
Clin_Feature_List="Key_Clin_features_with_PI_list.txt", surv_time="OS_month",
surv_event="OS")


# Model for univariate features
SurvPredPipe::MTLR_pred_model_f(train_clin_data = "Train_Clin.txt",
test_clin_data = "Test_Clin.txt", Model_type = 4, train_features_data =
"Train_Uni_sig_data.txt", test_features_data = "Test_Uni_sig_data.txt" ,
Clin_Feature_List="Key_univariate_features_list.txt",  surv_time="OS_month",
surv_event="OS")


# Model for Univariate + Clinical features
SurvPredPipe::MTLR_pred_model_f(train_clin_data = "Train_Clin.txt",
test_clin_data = "Test_Clin.txt", Model_type = 5, train_features_data =
"Train_Uni_sig_data.txt", test_features_data = "Test_Uni_sig_data.txt" ,
Clin_Feature_List="Key_univariate_features_with_Clin_list.txt", ,
surv_time="OS_month", surv_event="OS")
```

After, implementing `MTLR_pred_model_f` function , we got following outputs:

1. Model_with_PI.RData : Model on training data
2. survCurves_data.txt: Table containing predicted survival probability of each patient at different time points. This data can be further used to plot the survival curve of patients.

Let's check output of `MTLR_pred_model_f` function:

```
> # Load Survival curve data
> survCurves_data<- read.table("survCurves_data.txt", sep="\t", header=TRUE,
check.names=FALSE)

> #Check dimension of Survival curve data
> dim(survCurves_data)
[1] 15 19

> #View top 5 rows of Survival curve data
> head(survCurves_data,5)
  time_point TCGA-DH-A669-02 TCGA-WY-A859-01 TCGA-TQ-A7RN-01 TCGA-FG-A6IZ-01
TCGA-HW-8319-01 TCGA-TM-A7CF-02 TCGA-DU-7010-01 TCGA-HT-7686-01 TCGA-WY-A858-01
```

```
1   0.000000       1.0000000       1.0000000       1.0000000       1.0000000
1.0000000       1.0000000       1.0000000       1.0000000       1.0000000
2   3.466667       0.9909155       0.9972704       0.9773009       0.9955466
0.9960141       0.9961896       0.9905094       0.9939210       0.9959694
3   8.000000       0.9719876       0.9916419       0.9295280       0.9863267
0.9877695       0.9883111       0.9707273       0.9813015       0.9876316
4  14.000000       0.9514804       0.9852768       0.8798802       0.9760716
0.9785641       0.9795008       0.9493315       0.9674200       0.9783257
5  18.000000       0.9398923       0.9814651       0.8533639       0.9700671
0.9731460       0.9743045       0.9372695       0.9594131       0.9728513
  TCGA-TQ-A8XE-01 TCGA-DU-6408-01 TCGA-HT-7874-01 TCGA-DU-8162-01
TCGA-KT-A7W1-01 TCGA-VV-A829-01 TCGA-VM-A8CA-01 TCGA-HT-A61C-01 TCGA-S9-A6WN-01
1       1.0000000       1.0000000       1.0000000       1.0000000
1.0000000       1.0000000       1.0000000       1.0000000       1.0000000
2       0.9921240       0.9962126       0.9966324       0.9942035
0.9683777       0.9979307       0.9971536       0.9765579       0.9931295
3       0.9757360       0.9883821       0.9896764       0.9821754
0.9014997       0.9936732       0.9912822       0.9271999       0.9788516
4       0.9578822       0.9796236       0.9818651       0.9689217
0.8333833       0.9888145       0.9846516       0.8759960       0.9632163
5       0.9477176       0.9744564       0.9772317       0.9612589
0.7979822       0.9858691       0.9806885       0.8487143       0.9542526
```

3. mean_median_survival_time_data.txt: Table containing predicted mean and median survival time of each patient in the test data. This data can be further used for bar plots.

```
> #load mean_median_survival_time_data
> mean_median_survival_time_data<-
read.table("mean_median_survival_time_data.txt", sep="\t", header=TRUE,
check.names=FALSE)

> # Check dimension of mean_median_survival_time_data
> dim(mean_median_survival_time_data)
[1] 18  3

> #View top 5 rows of mean_median_survival_time_data
> head(mean_median_survival_time_data,5)
              IDs     Mean   Median
1 TCGA-DH-A669-02 67.84700 60.52883
2 TCGA-WY-A859-01 90.91569 85.75383
3 TCGA-TQ-A7RN-01 46.94777 36.31438
4 TCGA-FG-A6IZ-01 81.98710 81.61567
5 TCGA-HW-8319-01 84.05714 82.71186
```

4. Error_mat_for_Model.txt: Table containing performance parameters obtained on test data based on prediction model. It contains IBS score (Integrated Brier Score) =0.192, C-Index =0.81.

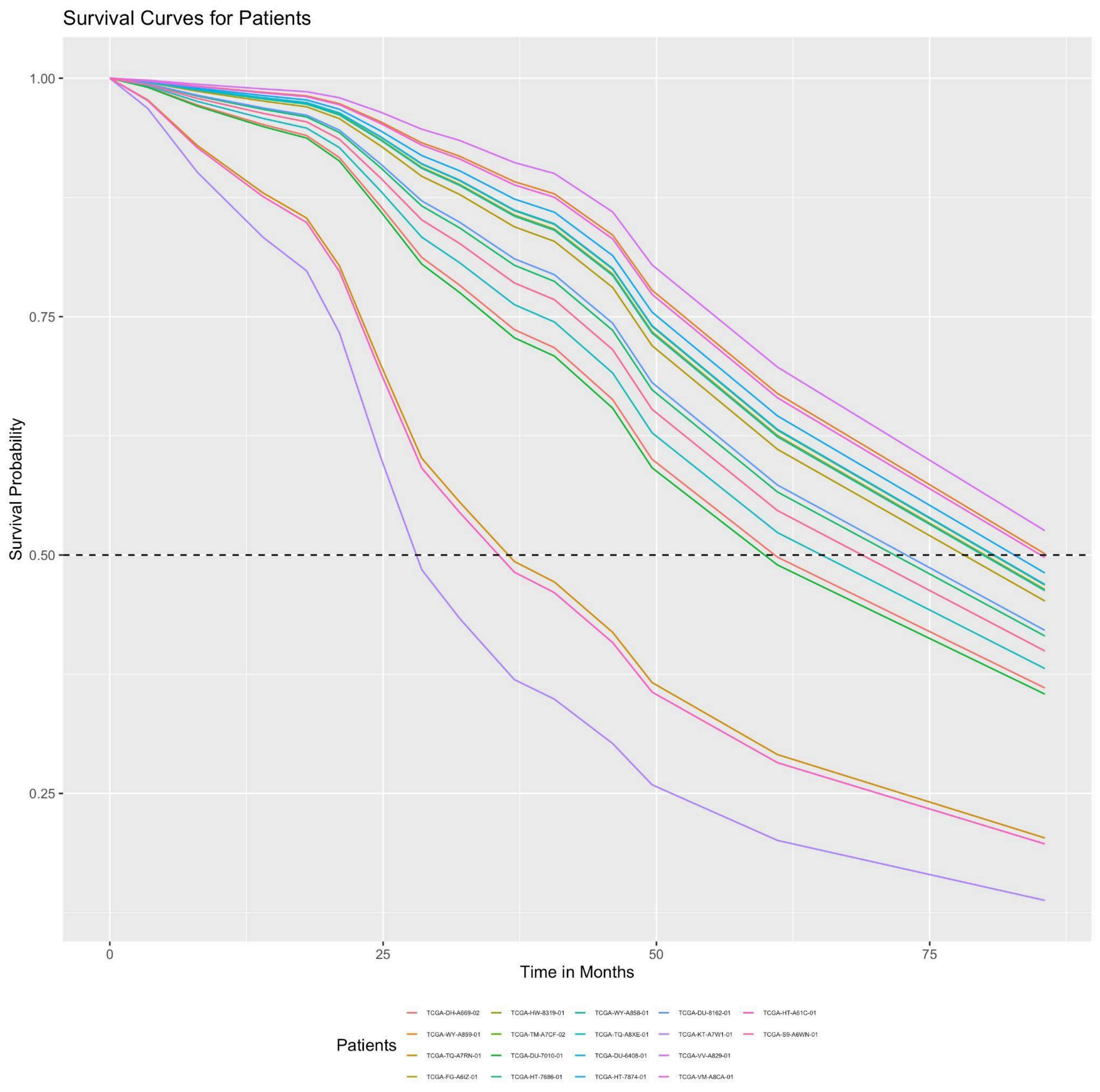
```
> #load Error mat file
> Error_mat_for_Model <- read.table("Error_mat_for_Model.txt", sep="\t",
header=TRUE,  check.names=FALSE)

> #View Error mat
> head(Error_mat_for_Model)
    IBS_Score c_index
IBS     0.192    0.81
```

Step 6 - Survival curves/plots for individual patient
Next to visualize survival of patients, we will plot survival curve plots using the `surv_curve_plots_f` function based on the data "`survCurves_data.txt`" that we obtained from the previous step after running the `MTLR_pred_model_f` function. Further, the surv_curve_plots_f function also allows highlighting a specific patient on the curve. Thus the function needs only two inputs: 1) Surv_curve_data, (2) Sample ID of a specific patient (e.g. **TCGA-TQ-A8XE-01**) that needs to be highlighted.

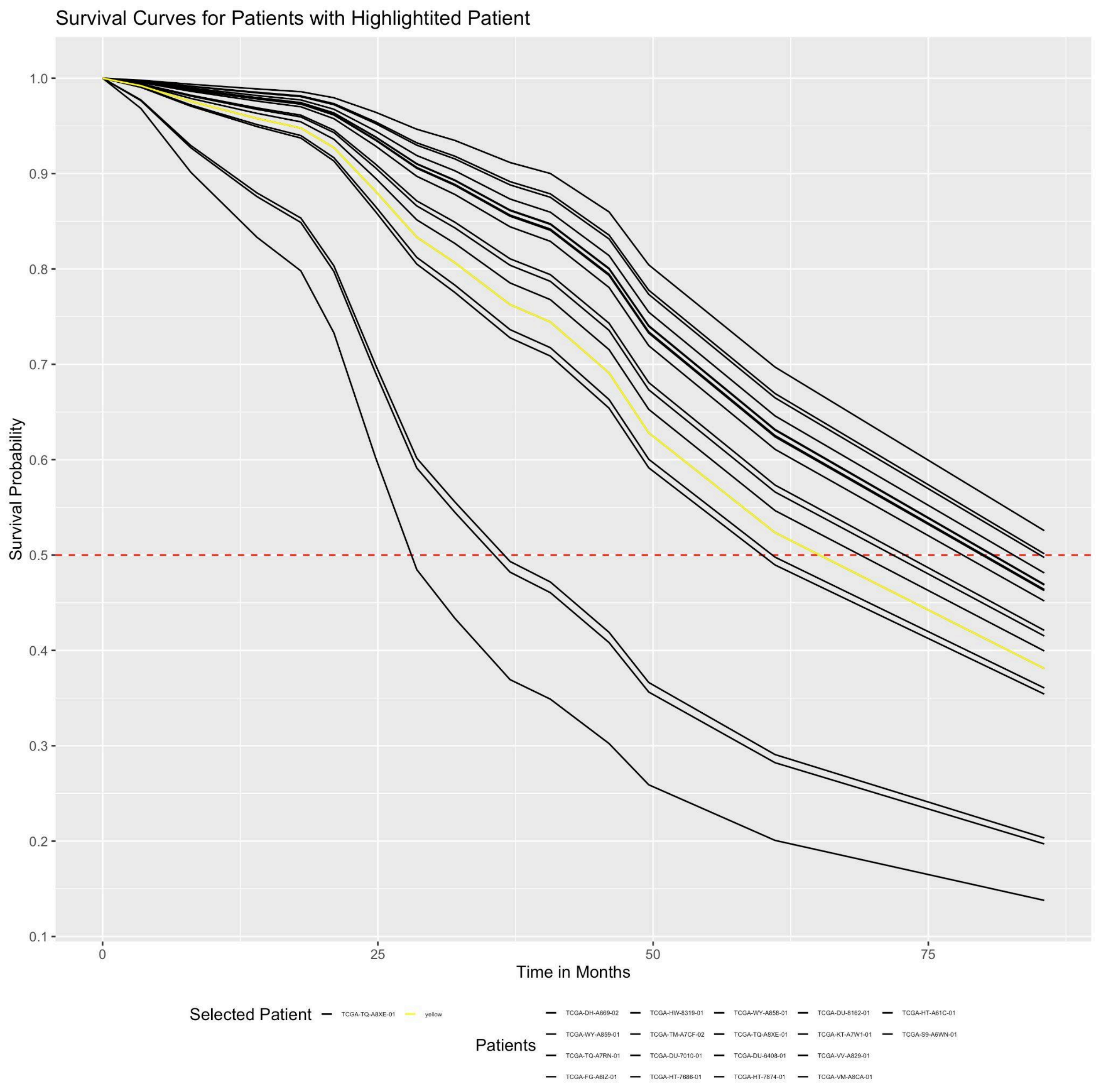
#Create Survival curves/plots for individual patients
```
SurvPredPipe::surv_curve_plots_f(Surv_curve_data="survCurves_data.txt",
selected_sample="TCGA-TQ-A8XE-01")
```

Let's check output of `surv_curve_plots_f` function:
Here, we obtained two output plots:

1. Survival curves for all patients in the test data with different colors



2. Survival curves for all patients (in black) and highlighted patient (yellow) in the test data



Step 7 - Bar Plot for predicted mean and median survival time of individual patients

Next to visualize predicted survival time of patients, we will plot barplot for mean/median using "`mean_median_surv_barplot_f`" function based on the data that we obtained from step 5 after running the MTLR_pred_model_f function. Further, the mean_median_surv_barplot_f function also allows highlighting a specific patient on the curve. Thus the function needs only two inputs: 1) surv_mean_med_data, (2) Sample ID of a specific patient (e.g. TCGA-TQ-A8XE-01) that needs to be highlighted.

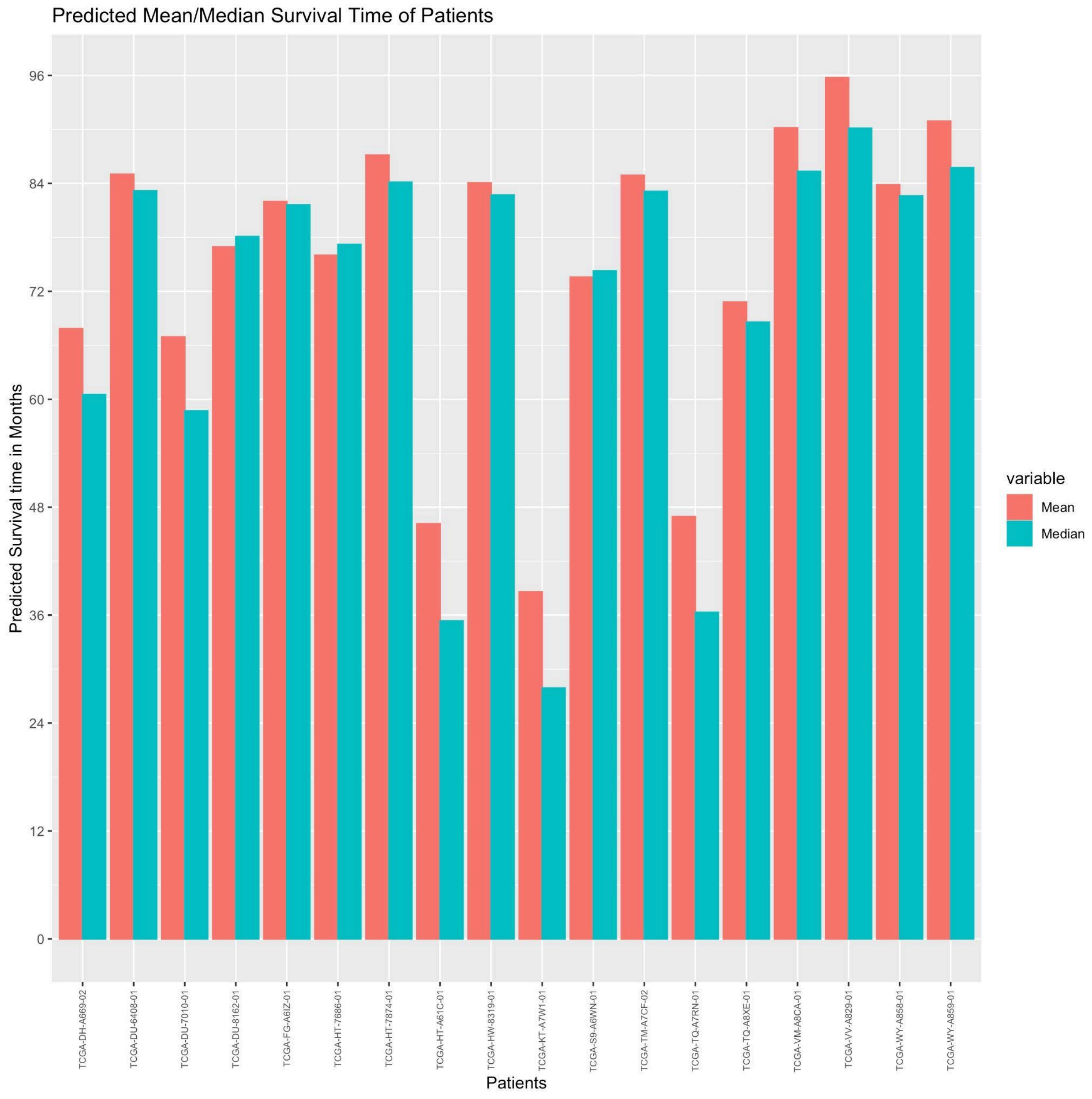
#Create Bar Plot for predicted mean and median survival time

```
SurvPredPipe::mean_median_surv_barplot_f(surv_mean_med_data="mean_median_surviv
al_time_data.txt", selected_sample="TCGA-TQ-A8XE-01")
```

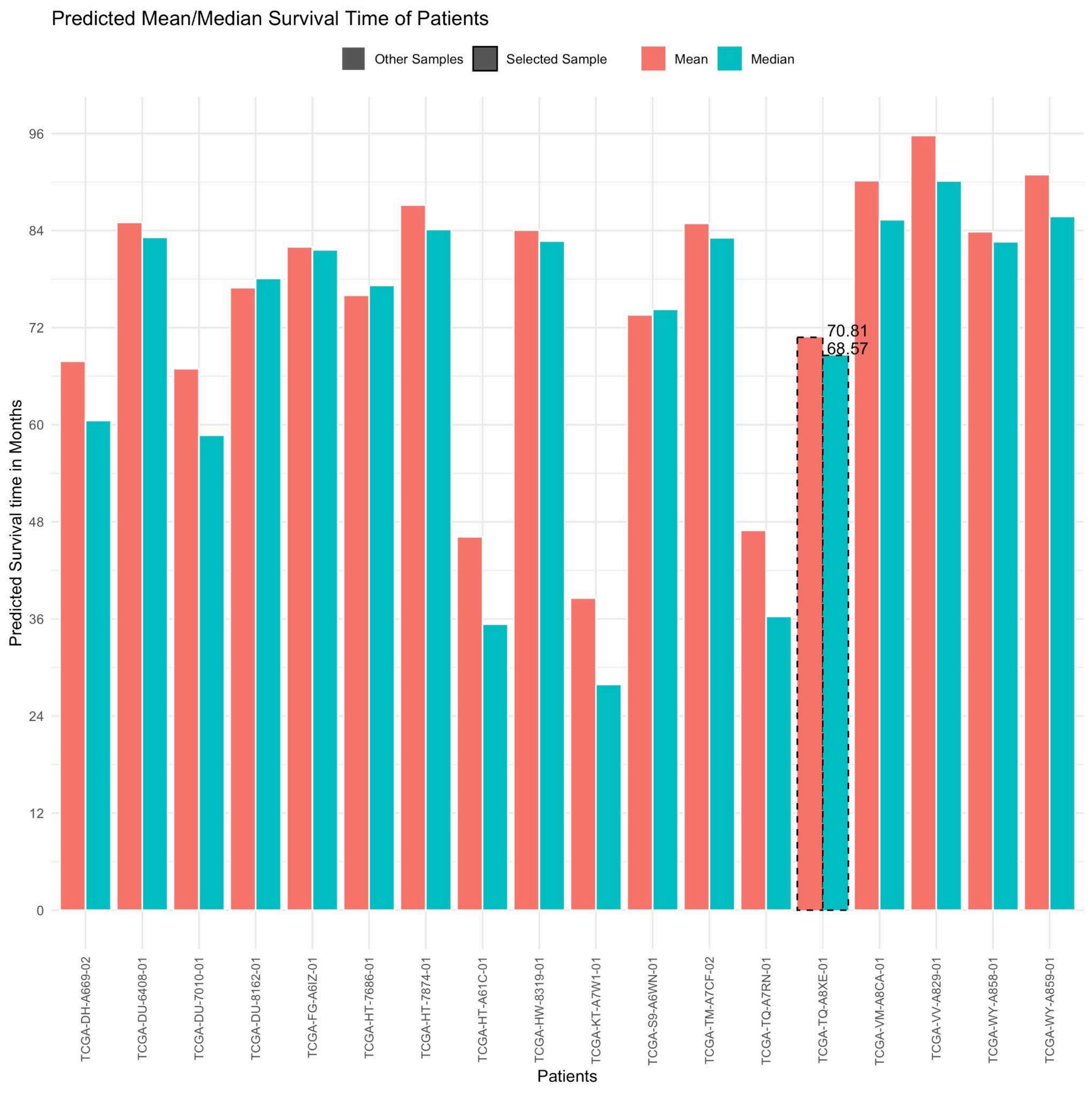
Let's check output of `mean_median_surv_barplot_f` function:

Here, we obtained two output plots:

1. Barplot for all patients in the test data, where the red color bar represents mean survival and cyan/green color bar represents median survival time.



2. Barplot for all patients with a highlighted patient (dashed black outline) in the test data. It shows this patient has a predicted mean and median survival is 81.58 and 75.50 months.



Step 8- Nomogram based on Key features

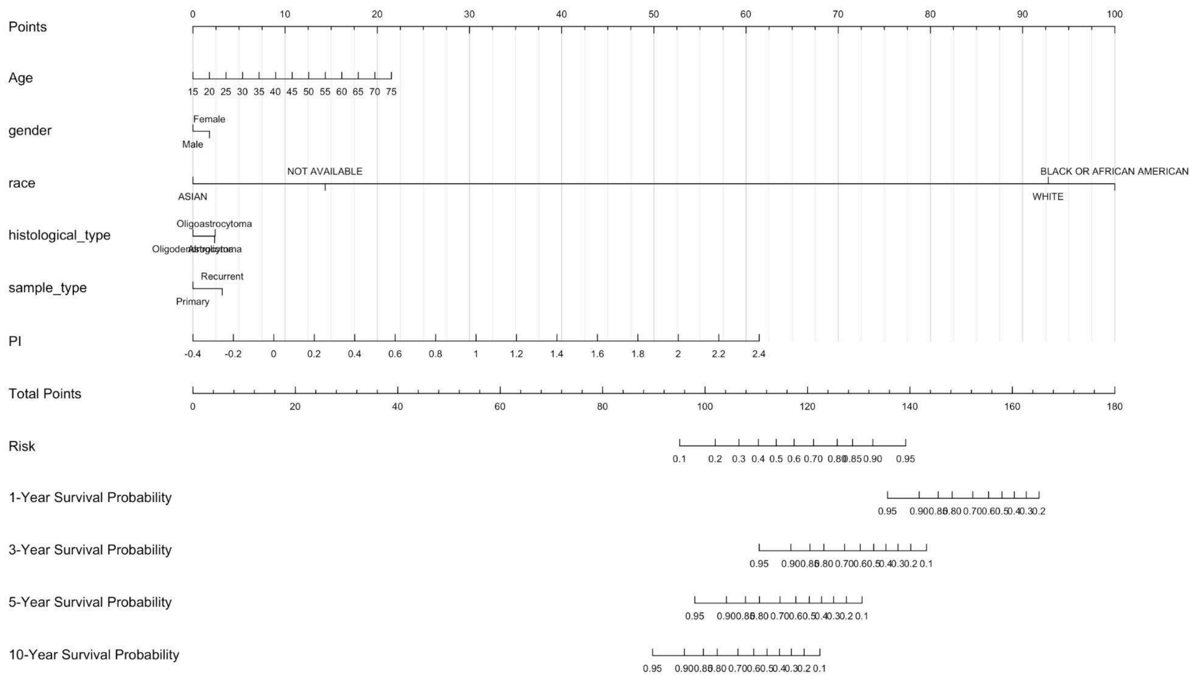
Next, the Nomogram_generate_f function of SurvPredPipe also provides an option to generate a nomogram plot based on user defined clinical and other features in the data. For instance, we will generate a nomogram based on 6 features (Age, gender, race, histological_type, sample_type, PI). Here, we will provide data containing all the features (Samples in rows and features in columns) (e.g. Train_Data_Nomogram_input.txt) and a list of features (feature_list_for_Nomogram.txt) based on which we want to generate a nomogram. Further, we also need to provide surv_time (name of column containing survival time in months, e.g. OS_month) and surv_event (name of column containing survival event information, e.g. OS) information in the data.

#Create Nomogram based on Key features

```
SurvPredPipe::Nomogram_generate_f(data="Train_Data_Nomogram_input.txt",
Feature_List="feature_list_for_Nomogram.txt", surv_time="OS_month",
surv_event="OS")
```

Let's check output of `Nomogram_generate_f` function:

Here, we will get a Nomogram based on features that we provide. This nomogram can predict Risk (Event risk, eg, Death), 1-year, 3-year, 5-year and 10 years survival of patients.


Points
Age
gender
Female
Male
race
NOT AVAILABLE
BLACK OR AFRICAN AMERICAN
ASIAN
WHITE
histological_type
Oligoastrocytoma
Oligodendroglioma
Astrocytoma
sample_type
Recurrent
Primary
PI
Total Points
Risk
1-Year Survival Probability
3-Year Survival Probability
5-Year Survival Probability
10-Year Survival Probability