SurvPredPipe

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

Package: SurvPredPipe
Title: R-Package for the Pipeline to Predict Survival Probability of Cancer Patients
Version: 1.0
Authors@R:
person("Harpreet", "Kaur", "harpreet.kaur2@nih.gov", "hks04180@gmail.com", role = c("aut", "cre"),
comment = c(ORCID = "https://orcid.org/0000-0003-0421-8341"))
Description: This package is about the 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..
License: GPL-2 | file LICENSE
Encoding: UTF-8
Roxygen: list(markdown = TRUE)
RoxygenNote: 7.3.1
Imports: Rcpp, MASS, dplyr, reshape2, glmnet, caret, survival, survminer, ggfortify , ggplot2, MTLR, SurvMetrics, pec, preprocessCore,rms, Hmisc, ROCR
Suggests:
knitr,
rmarkdown,
testthat (>= 3.0.0)
VignetteBuilder: knitr
Config/testthat/edition: 3

Index

Introduction

The S**urvPredPipe** package is about the 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.

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

---

Installation

Steps for Installation on Local system from Github Repository:

Install remote package
install.packages("remotes")
load remotes package
library("remotes")
Install SurvPredPipe package
remotes::install_github("hks5august/SurvPredPipe", local = TRUE)
Load package
library("SurvPredPipe")

---

data_process_f : Data processing function

Description
This function converting OS time (in days) into months and then removing samples where OS time information is missing

Usage

```
data_process_f(data,col_num, surv_time , output)
```

Arguments
data: data (Patients data with clinical and gene expression, where samples are in rows and features/genes are in columns)
col_num: column number in data at where clinical info ends

surv_time: column name which contain survival time (in days) information
output : name of output file, in which the user wants to store data. In this output file, there is an additional column in the data with OS time information in terms of OS_month (where days are converted into months), and any samples with missing OS time information will be excluded.

Details
This function converts survival time (e.g. OS time) which is present in days in the data into months and then removes samples where survival (OS/OS.time) information is missing. Here, the user needs to provide input data in tsv or txt format, where samples must be in rows and features must be in columns. Further, user needs to provide ***col_num***, i.e. column number at which clinical/demographic and survival information ends (e.g. 20), ***surv_time***, i.e. name of column which contain survival time (in days) information (e.g. OS.time) and ***output*** file name to store output data (e.g. **New_data.txt**). After data processing, the user will get a new output file "**New_data.txt**". The `data_process_f` function will remove samples where survival information is missing in the data. Besides, a new column will be added in the data with column name "OS_month" where OS time is available in months.

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

---

tr_test_f : Train - test split Function

Description
This function split data into training and test data as per user defined ratio or fraction

Usage
```
tr_test_f(data,fraction, train_data test_data)
```

Arguments
data: a data frame where features in the columns and samples must be in rows
fraction: Fraction by which user want to split data for training data; e.g. 90% training, so fraction would be 0.9
train_data: name of training set that user can provide
test_data: name of test set that user can provide

Details

This function splits Data into Training and Test Subsets. Since, the user needs training and test datasets for the purpose of feature selection and model development. This function will use output from the previous function `data_process_f` as an input ***data*** (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, user also need to provide ***train_data, test_data*** output names to store training and test set output(e.g. **train_FPKM.txt, test_FPKM.txt**). Thus, the ***tr_test_f*** function will give two new outputs: **train_FPKM.txt, test_FPKM.txt**.

Example

```
tr_test_f(data="New_data.txt",fraction=0.9,
train_data="train_FPKM.txt", test_data="test_FPKM.txt")
```

---

train_test_normalization_f: Data normalization Function

Description

This function first tranform FPKM data into log2-scale transformation, followed by quantile normalization, where training data values used as target matrix both for training and test data for quantile normalization

Usage

```
train_test_normalization_f(train_data,test_data, col_num,
train_clin_data, test_clin_data,
train_Normalized_data_clin_data, test_Normalized_data_clin_data)
```

Arguments

train_data : training data (TSV) (Patients data with clinical and gene expression, where samples are in rows and features/genes are in columns)
test_data_data : test data (TSV) (Patients data with clinical and gene expression, where samples are in rows and features/genes are in columns)
col_num : column number in data at where clinical info ends
train_clin_data: name of training data output which stores only clinical information
test_clin_data: name of test data output which stores only clinical information
train_Normalized_data_clin_data: output file name - outfile file containing training clinical data and normalized data (which is log-scaled followed by quantile normalized data).
test_Normalized_data_clin_data: output filename - outfile file containing test clinical data and normalized data (which is log-scaled followed by quantile normalized data).

Details

This function performs data normalization. Since, expression is available in terms of FPKM values. Thus, the `train_test_normalization_f` function will first convert FPKM value into log scale transformation [log2(FPKM+1) followed by quantile normalization using the "`preprocessCore`" package (DOI: 10.18129/B9.bioc.preprocessCore). The training data will be used as a target matrix for quantile normalization. Here, the user needs to provide ***train_data*** and ***test_data*** (that we obtained from the previous train-test split function ***tr_test_f***). Further, the user needs to provide **col_num** i.e. column number where clinical information ends (e.g. 21) in the input datasets. Furthermore, the user 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). After, running the `train_test_normalization_f` function will give 4 outputs: ***Train_Clin.txt*** - Contains only Clinical features, ***Test_Clin.txt***- contains only Clinical features of Test samples; ***Train_Norm_data.txt***- Clinical features with normalized values of genes for training samples; ***Test_Norm_data.txt*** - Clinical features with normalized values of genes for test samples.

Example

```
train_test_normalization_f(train_data="train_FPKM.txt",test_data
="test_FPKM.txt", col_num=21, train_clin_data="Train_Clin.txt",
test_clin_data="TestClin.txt",
train_Normalized_data_clin_data="Train_Norm_data.txt",
test_Normalized_data_clin_data="Test_Norm_data.txt")
```

Univariate_sig_features_f : Univariate feature selection Function

---

Description

This function selects significant features (p-value <0.05) based on median expression cut off values of features using Univariate Survival analysis.

Usage

```
Univariate_sig_features_f(train_data, test_data, col_num,
surv_time, surv_event, output_univariate_train,
output_univariate_test)

Univariate_sig_features_f(train_data=args[1], test_data=args[2],
col_num=as.numeric(args[3]), surv_time=as.numeric(args[4]) ,
surv_event=as.numeric(args[5]) ,output_univariate_train=args[6],
output_univariate_test=args[7])
```

Arguments

train_data : Normalized training data (Patients data with clinical and gene expression, where samples are in rows and features/genes are in columns)
test_data : Normalized test data (Patients data with clinical and gene expression, where samples are in rows and features/genes are in columns)
col_num : column number in data at where clinical info ends
surv_time: column name which contain survival time (in days) information
surv_event: column name which contain survival event information
output_univariate_train : name of output to store univariate selected features for training data
output_univariate_test : name of output to store univariate selected features for test data

Example

```
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")
```

Details

This function selects 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, `survfit` and `coxph` functions of `survival` and `survminer` packages (ref) were used to compute beta coefficient, Hazard ratio (HR), concordance index to assess the performance of features. Further, a log-rank test was used to compute significance in terms of p-value. Here, the user needs to provide Normalized training (e.g.Train_Norm_data.txt) and test data (e.g. Test_Norm_data.txt) as input data that can be 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.
Thus, Univariate_sig_features_f 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.
2. **Train_Uni_sig_data.txt:** It contains expression of significant genes selected by univariate survival analysis for training samples.
3. **Test_Uni_sig_data.txt:** It contains expression of significant genes selected by univariate survival analysis for test samples.

Lasso_PI_scores_f LASSO Prognostic Index Calculator function

Description

This function will create a PI (Prognostic Index) score based on selected LASSO genes for training and test data. Here, firstly, LASSO will select genes with beta coefficient values based on COX lasso regression using 5-fold cross validation. Subsequently, PI score will be calculated by multiplying expression of genes with their beta coeff values.

Usage

```
SurvPredPipe::Lasso_PI_scores_f(train_data,test_data, nfolds,
col_num, surv_time , surv_event, train_PI_data, test_PI_data)
```

Arguments

train_data: training normalized data (Patients data with clinical and gene expression, where samples are in rows and features/genes are in columns)
test_data :test normalized data (Patients data with clinical and gene expression, where samples are in rows and features/genes are in columns)
nfolds : Number of folds for cross in cvglmnet model to select top features
col_num : column number in data at where clinical info ends
surv_time: column name which contain survival time (in days) information
surv_event: column name which contain survival event information
train_PI_data : name of output file containing LASSO selected genes and PI score data for training data
test_PI_data : name of output file containing LASSO selected genes and PI score data for test data

Details

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. Here, we used the `glmnet` package to perform LASSO regression to select top features with their beta coefficient. For instance, the top 5 features selected by LASSO include G1, G2, G3, G4, and G5 and their beta coefficient values are B1, B2, B3, B4, and B5. 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 end (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.

Thus, 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
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.

Example:

```
SurvPredPipe::Lasso_PI_scores_f(train_data="Train_Norm_data.txt"
,test_data="Test_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" )
```

MTLR_pred_model_f **MTLR prediction model function**

Description

This function can generate 5 Prediction models types based on :(1) Clinical features, (2) PI score, (3) PI score + Clin, (4) Significant Univariate features, (5) Significant Univariate features + clin features, using MTLR method. Furthermore, this model will predict the survival of test data in terms of survival probability over different time points, mean and median survival time of patients in test data.

Usage

```
MTLR_pred_model_f(train_clin_data , test_clin_data , Model_type,
train_features_data, test_features_data , Clin_Feature_List,
surv_time, surv_event)
```

Arguments

train_clin_data : training data with Clin features (Patients data with clinical and gene expression, where samples are in rows and features/genes are in columns)
test_clin_data : test data with Clin features (Patients data with clinical and gene expression, where samples are in rows and features/genes are in columns)
Model_type: Prediction Model type : 1 - Clinical features, 2 - PI score features, 3- PI score + Clinical features ; 4 - Univariate features, 5 - Univariate features + Clinical features
train_features_data : Training data with selected features only (Significant Univariate/LASSO PI score)
test_features_data : Training data with selected features only (Significant Univariate/LASSO PI score)
Clin_Feature_List : A list of feature containing their name to build model
surv_time: column name which contain survival time (in days) information
surv_event: column name which contain survival event information

Details

After selecting significant or key features using LASSO or Univariate survival analysis; `MTLR_pred_model_f` will develop a prediction model to predict survival probability of patients. In the `MTLR_pred_model_f` function, we used the `MTLR` package to develop a prediction model to predict survival probability, mean and median survival time of individual patients. Then, `SurvMetrics` and `pec` were used to compute Concordance Index and IBS (Integrated Brier Score) of the test data. The `MTLR_pred_model_f` function gives the user

five 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. For instance the user is interested in developing a model based on PI score. Thus, the user needs 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, the user 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

OUTPUTs:
Model_with_PI.RData: Model will be saved.
survCurves_data.txt: Text file containing survival curve data
Mean_median_survival_time_data.txt: Text file containing mean and median survival time data.
survival_result_based_on_MTLR.txt:
Error_mat_for_Model.txt: Text file containing IBS score and C-index values.

Example:

```
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 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")
```

---

surv_curve_plots_f **Survival Curve plot function**

Description

This function generates individual survival curves for patients. Moreover, user can also highlight one specific sample by providing sample IDs

Usage:

```
surv_curve_plots_f(Surv_curve_data,  selected_sample)
```

Arguments

Surv_curve_data : Survival probability data for survival curve of patients

selected_sample: ID of a specific sample user wants to highlights on the plot against the background of curves of other samples

Details

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.

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

Example:

```
surv_curve_plots_f(Surv_curve_data="survCurves_data.txt",
selected_sample="TCGA-TQ-A8XE-01")
```

---

mean_median_surv_barplot_f: Survival time barplot function

Description

This function generates barplots for predicted mean and median survival time of patients. Besides, user can also highlight one specific sample by providing sample IDs

Usage:

```
mean_median_surv_barplot_f( surv_mean_med_data=args[1],
selected_sample=as.character(args[2]) )
```

Arguments:

Surv_curve_data :args1 - Predicted Mean median survival time data for patients
selected_sample:args2 - ID of a specific sample user wants to highlights on the plot against the background of curves of other samples

Details

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.

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.

Example:

```
mean_median_surv_barplot_f(surv_mean_med_data="mean_median_survi
val_time_data.txt", selected_sample="TCGA-TQ-A8XE-01")
```

Nomogram_generate_f :Nomogram Generate function

Description

This function generates a nomogram based on a user provided list of features. It

Usage:

```
Nomogram_generate_f(data= args[1],  Feature_List=args[2],
surv_time=as.numeric(args[3]), surv_event=as.numeric(args[4]))
```

Arguments

data: data (.txt format), where features should be in columns and samples should be in rows
Feature_List: List of features based on which user want to develop nomogram
surv_time: column name which contain survival time (in days) information
surv_event: column name which contain survival event information

Details

The `Nomogram_generate_f` function of SurvPredPipe provides an option to generate a nomogram plot based on user defined clinical and other features in the data. Here, we are using the `coxph` function of the `rms` package to generate a nomogram. For instance, if the user is interested in generating a nomogram based on 6 features (Age, gender, race, histological_type, sample_type, PI). Then, the user needs to 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 the user can 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. Thus, the user 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.

Example

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