Package 'SPreFuGED'

July 29, 2016

Type Package

Title Selecting a Predictive Function for a Given Gene Expression Data

Version 1.0

Date 2016-07-27

Author Victor L Jong, Kit CB Roes & Marinus JC Eijkemans

Maintainer Victor L Jong <v.l.jong@umcutrecht.nl>

Description The recent advancement of high-throughput technologies has led to frequent utilization of gene expression and other ``omics" data for toxicological, diagnostic or prognostic studies in and clinical applications. Unlike in classical predictions where the number of samples is greater than the number of variables (n>p), the challenge faced with prediction using ``omics" data is that the number of parameters greatly exceeds the number of samples (p>>n). To solve this curse of dimensionality problem, several predictive functions have been proposed for direct and probabilistic classification and survival predictions. Nevertheless, these predictive functions have been shown to perform differently across datasets. Comparing predictive functions and choosing the best is computationally intensive and leads to selection bias. Thus, the question which function should one choose for a given dataset is to be ascertained. This package implements the approach proposed by Jong et al., (2016) to address this question.

Depends R(>= 3.2.4)

Imports stats, methods, utils, Biobase, CMA, lme4, lattice, limma, boot, mvtnorm

Suggests e1071, MASS, class, nnet, glmnet, randomForest

License GPL (>= 2)

LazyData True

NeedsCompilation no

Repository CRAN

Date/Publication 2016-07-29 12:35:58

R topics documented:

`SPreFuGED-package` *Selecting a Predictive Function for a Given Gene Expression Data*

Description

The recent advancement of high-throughput technologies has led to frequent utilization of gene expression and other "omics" data for toxicological, diagnostic or prognostic studies in and clinical applications. Unlike in classical predictions where the number of samples is greater than the number of variables (n>p), the challenge faced with prediction using "omics" data is that the number of parameters greatly exceeds the number of samples (p»n). To solve this curse of dimensionality problem, several predictive functions have been proposed for direct and probabilistic classification and survival predictions. Nevertheless, these predictive functions have been shown to perform differently across datasets. Comparing predictive functions and choosing the best is computationally intensive and leads to selection bias. Thus, the question which function should one choose for a given dataset is to be ascertained. This package implements the approach proposed by Jong et al., (2016) to address this question.

Details

The package allows one to determine an optimal predictive function among several functions for either binary direct classification, binary probabilistic classification or survival prediction for a given gene expression data. It also presents an interface to simulate gene expression data and to compare classification survival prediction functions on a given data. The most important workflow of this package is as follows:

1. Estimate the gene expression data characteristics using estimateDataCha
2. Fit a specific linear mixed effects model using fitLMEModel
3. Predict the performance of the functions on the given data using SPreFu
4. Plot the results of step 3 using plotSPreFu.
5. Other functions are(were) used for simulation studies presented in most of the references.

Author(s)

Victor L Jong, Kit CB Roes & Marinus JC Eijkemans

Maintainer: Victor L Jong <v.l.jong@umcutrecht.nl>

References

Jong VL, Novianti PW, Roes KCB & Eijkemans MJC. Selecting a classification function for class prediction with gene expression data. Bioinformatics (2016) 32(12): 1814-1822;

---

avAcc *Average accuracy data*

Description

A data frame of 11520 data points (rows) corresponding to the 1152 different simulation scenarios by 10 classification functions. The studied data variables and the average misclassification error rates (acc) of the classification functions on the columns.

Usage

```
data("avAcc")
```

Format

A data frame with 11520 observations on the following 8 variables.

`class` a character vector of the predictive functions

`variance` a numeric vector of the different values of the variance parameter

`deCorr` a numeric vector of different pairwise correlation values within informative genes

`propDE` a numeric vector different values of the proportion of informative genes

`log2FC` a numeric vector of different values of absolute log2 fold changes

`sampSize` a numeric vector of different values of absolute sample sizes

`otherCorr` a numeric vector of the different values of the parameter for the pairwise correlations within non-informative and/or between non-informative and informative genes

`acc` a numeric vector of misclassification error rates

Details

This data contains the different values of the six studied factors and the error rates for the different fucntions in different simulation scenarios. And it is required for binary direct classification, to fit the LME model necessary to predict optimal function(s) for a given gene expression dataset.

Source

Jong VL, Novianti PW, Roes KCB & Eijkemans MJC. Selecting a classification function for class prediction with gene expression data. Bioinformatics (2016) 32(12): 1814-1822

Examples

```
data(avAcc)
## maybe str(avAcc) ; plot(avAcc) ...
```

`covMat` *A covariance matrix generating function.*

Description

For a given number of genes and a proportion of differentially expressed (informative) genes, the this function creates a covariance matrix by sampling variances from an exponential distribution with lambda and the correlation values corrDE and corrOther. Where corrOther is generated from a normal distribution with mean=0, and sigma as SD.

Usage

```
covMat(pAll = 1000, pie = 0.05, lambda, corrDE, sigma)
```

Arguments

`pAll`	the total number of genes (Default is 1000). For desktop users, we encourage pAll <=2500 for compuational reasons.
`pie`	a value in the interval (0, 1) and corresponds to the proportion of differentially expressed (informative) genes (Defualt is 0.05)
`lambda`	a positive rate parameter for sampling variances from an exponential distribution. The smaller the value the larger the variances.
`corrDE`	a value in the interval [0, 1] specifying the correlation values of DE genes to each other. Half of which are up-regulated (positively associated to survival time) and the others are down-regulated (negatively associated to survival time). The inter-cluster (between up- and down-regulated) genes take negatively signed value of corrDE. The value 0 corresponds to complete independence of these DE genes.
`sigma`	a value in the interval [0, 1] specifying the distribution of correlations within noisy genes and between noisy genes and informative genes. Where 0 means complete indipendence of noisy genes to each other and to informative genes.

Details

This functions assumes three clusters of genes (up-regulated, down-regulated and noisy genes). While the pairwise correlations of the DE genes is a descrete value corrDE, the correlations of the non-DE genes are sampled from a normal distribution with mean zero and SD=sigma. Values beyond the interval [-1, 1] are unformly converted to that interval.

Value

A list containing:

`cov`	the covariance matrix generated
`pie`	the proportion of differentially expressed genes

Author(s)

Victor Lih Jong

References

Jong VL, Novianti PW, Roes KCB & Eijkemans MJC. Selecting a classification function for class prediction with gene expression data. Bioinformatics (2016) 32(12): 1814-1822

See Also

generateGED

Examples

```
myCov<-covMat(pAll=100, lambda=2, corrDE=0.75, sigma=0.25);
#Observe the covariance matrix of 6 genes, 2 each from up-regulated, down-regulated and non-DE
myCov$cov[c(1,2, 4,5, 30,31), c(1,2, 4,5, 30,31)];
myCov$pie;
```

`directClass` *A function to build and evaluate 10 different classification functions on a given gene expression data.*

Description

This function uses either simulated (train and test) or real-life gene expression data to build and evaluate binary direct classifiers with 10 different classification functions [LDA, KNN, NNET, PAM, QDA, RF, Ridge (PLR2), Lasso (PLR1), Elastic Net (PLR12) and SVM] by minimizing the misclassification error rates.

Usage

```
directClass(data, dataY = NULL, simulated = TRUE, fold = 5)
```

Arguments

`data`	an object returned by generateGED or a matrix of expression values containing genes in the rows and samples in the columns.
`dataY`	an optional vector of class lables that can be coerced to a factor. Must be supplied when the data argument is not simulated data.
`simulated`	logical, indicating whether the data supplied is simulated data (Default is TRUE).
`fold`	the number of cross-validation times to divide the real-life data into 2/3 train and 1/3 test with stratisfication(Default is 5). For meaningful comparison we recommend fold=100.

Details

See reference for detail on which classification functions and/or parameters are optimized in this function and how the classifiers are built and evaluated. Please note that for large datasets and large values of "niter" from generateGED or fold, this function might take quite sometime. Of course, if it takes time to train a single function, what more of training 10 functions at once?

Value

An Lx10 matrix of misclassification error rates. Where L is the number of iterations (niter) when the data is simulated data from generateGED or the number of folds (fold=L) when the data is real-life data and 10 are the number of classification functions.

Author(s)

Victor Lih Jong

References

Jong VL, Novianti PW, Roes KCB & Eijkemans MJC. Selecting a classification function for class prediction with gene expression data. Bioinformatics (2016) 32(12): 1814-1822

See Also

`covMat`, `generateGED` and `plotDirectClass`

Examples

```
#Let us use simulated data build the 10 classification functions
myCov<-covMat(pAll=100, lambda=2, corrDE=0.75, sigma=0.25);
myData<-generateGED(covAll=myCov, nTrain=30, nTest=10);
myClassResultsSimulatedData<-directClass(data=myData); #Takes roughly 60 Sec
```

---

estimateDataCha *A function to estimate data characteristics*

Description

This function fits limma models (or univariate Cox's models t) to determine DE (informative) genes and then computes the proportion of DE (informative) genes, log2FC (coefficients or betas), pairwise correlation of DE (informative) and noisy genes, genes' variances, sample sizes and proportion of events (for survival data).

Usage

```
estimateDataCha(data, dataY, type = "Binary")
```

Arguments

data	a matrix of expression values with rows corresponding to genes and columns to samples
dataY	a binary vector of class labels or a survival outcome as produced by Surv. Its length must be equal to the number of columns of data.
type	takes Binary(Default) or Survival as values and correspond to binary classification or survival prediction

Details

At the moment, only binary classification has been implemented.

Value

A 1x7 (for Binary) or 1x8 (for Survival) matrix containing the estimates (row) of the data characteristics (columns)

Author(s)

Victor Lih Jong

References

Jong VL, Novianti PW, Roes KCB & Eijkemans MJC. Selecting a classification function for class prediction with gene expression data. Bioinformatics (2016) 32(12): 1814-1822

See Also

fitLMEModel, SPreFu and plotSPreFu

Examples

```
#Let us consider a single simulated train data as our real-life dataset
myCov<-covMat(pAll=100, lambda=2, corrDE=0.75, sigma=0.25);
myData<-generateGED(covAll=myCov, nTrain=30, nTest=10);
data<-myData[[1]]$trainData;
dataY<-myData[[1]]$trainLabels;
myDataCha<-estimateDataCha(data, dataY);
```

`fitLMEModel` *A function to fit a linear mixed effects (LME) model of performance measure on the studied variables*

Description

This function fits a LME model to the log-odds of accuracy (for binary direct classifiers), logit transformation of the transformed Brier Score (for binary probabilistic classifiers) or logit transformation of the transformed integrated Brier score (for survival data).

Usage

```
fitLMEModel(type="Accuracy")
```

Arguments

`type`	takes Accuracy (Default), for direct classifiers or Probability, for probabilistic classifiers or Survival, for survival predictions.

Details

Depending of the value of type, this function uses either avAcc, avBS or avSurv data to build a LME model. Only the avAcc is available and hence LME model of log-odds accuracy is possible at the moment.

Value

A list containing:

`model`	an object of class "lmer" for which several fucntions can be applied
`type`	the type of predictions (Accuracy, Probability or Survival)
`fitData`	fitted data, contains the variables and their standardized values

Author(s)

Victor Lih Jong

References

Jong VL, Novianti PW, Roes KCB & Eijkemans MJC. Selecting a classification function for class prediction with gene expression data. Bioinformatics (2016) 32(12): 1814-1822

See Also

`estimateDataCha`, `SPreFu` and `plotSPreFu`

Examples

```
myFit<-fitLMEModel();  #Takes roughly 250 Sec
```

generateGED *A function to simulate two groups gene expression data (GED)*

Description

This function draws two sets of vectors (train and test samples' labels) from a binomial distribution and generates two gene expression datasets (train and test data) from a multivariate normal distribution with a mean vector U[6,10] and a given within-class covariance matrix, at each iteration.

Usage

```
generateGED(covAll, nTrain, nTest, log2FC = 1, niter = 3, prob = 0.5)
```

Arguments

covAll	an object returned by covMax or a list containing a covariance matrix cov and the proportion of DE genes pie.
nTrain	the number of samples in the training set
nTest	the number of samples in the test set
log2FC	the absolute Log2 fold changes (effect sizes) for DE genes (Default is 1)
niter	the number of iterations (train/test datasets to be generated). Default is 3
prob	the probability of success for the binomial sampling. Default is 0.5

Value

A list of length niter. Each element of which is a list containing:

trainData	a matrix of the training data
trainLabels	a binary vector of class labels of the training samples
testData	a matrix of the test data
testLabels	a binary vector of class labels of the test samples

Author(s)

Victor Lih Jong

References

Jong VL, Novianti PW, Roes KCB & Eijkemans MJC. Selecting a classification function for class prediction with gene expression data. Bioinformatics (2016) 32(12): 1814-1822

See Also

covMat, directClass and plotDirectClass

Examples

```
myCov<-covMat(pAll=100, lambda=2, corrDE=0.75, sigma=0.25);
myData<-generateGED(covAll=myCov, nTrain=30, nTest=10);
myFirstTrainData<-myData[[1]]$trainData; myFirstTrainLabels<-myData[[1]]$trainLabels;
myFirstTestData<-myData[[1]]$testData; myFirstTestLabels<-myData[[1]]$testLabels;
```

`plotDirectClass` *A plotting function for the performance (Accuracy) of direct classifiers*

Description

This function produces boxplots of the expected accuracies of each classification function from call to directClass.

Usage

```
plotDirectClass(restDirectClass)
```

Arguments

`restDirectClass`
: an object returned by directClass.

Value

A plot of classification funcions and their expected accuracies

Author(s)

Victor Lih Jong

References

Jong VL, Novianti PW, Roes KCB & Eijkemans MJC. Selecting a classification function for class prediction with gene expression data. Bioinformatics (2016) 32(12): 1814-1822;

See Also

`covMat`, `generateGED` and `directClass`

Examples

```
#Let us use simulated data build the 10 classification functions
myCov<-covMat(pAll=100, lambda=2, corrDE=0.75, sigma=0.25);
myData<-generateGED(covAll=myCov, nTrain=30, nTest=10);
myClassResultsSimulatedData<-directClass(data=myData); #Takes roughly 60 Sec
plotDirectClass(myClassResultsSimulatedData);
```

plotSPreFu *A plotting function of the predicted performance of classification functions on a given dataset*

Description

This function plots the predicted accuraces, Brier scores or Hazard ratios for each predictive function on a given dataset, as predicted by SPreFu.

Usage

```
plotSPreFu(restSPreFu)
```

Arguments

restSPreFu	an object returned by SPreFu

Value

A plot of predictive funcions and their expected performance (Accuracy, Brier score or Integrated Brier score)

Author(s)

Victor Lih Jong

References

Jong VL, Novianti PW, Roes KCB & Eijkemans MJC. Selecting a classification function for class prediction with gene expression data. Bioinformatics (2016) 32(12): 1814-1822;

See Also

estimateDataCha, fitLMEModel and SPreFu

Examples

```
#Let us consider a single simulated train data as our real-life dataset
myCov<-covMat(pAll=100, lambda=2, corrDE=0.75, sigma=0.25);
myData<-generateGED(covAll=myCov, nTrain=30, nTest=10);
data<-myData[[1]]$trainData;
dataY<-myData[[1]]$trainLabels;
myDataCha<-estimateDataCha(data, dataY);
myFit<-fitLMEModel();  #Takes roughly 250 Sec
myPred<-SPreFu(myDataCha, myFit);
plotSPreFu(myPred);
```

SPreFu *A function for selecting an optimal predictive function for a given gene expression data.*

Description

This function uses the LME model and the estimated data characteristics to predict the accuracy (for binary direct classifiers) or transformed Brier score (for binary probabilistic classifiers) or transformed integrated Brier scores (for survival predictions).

Usage

```
SPreFu(dataCha, restModel)
```

Arguments

dataCha	an object returned by estimateDataCha and contains estimates of the data characteristics.
restModel	an object returned by fitLMEModel and contains the fitted LME model to be used for predictions.

Value

A list containing:

dataCha	a data frame of the estimated data characteristics, the predictive functions and their predicted performance
type	is the type of prediction (Accuracy, Probability or Survival) and determines what kind of plots to be produced by plotSPreFu

Author(s)

Victor Lih Jong

References

Jong VL, Novianti PW, Roes KCB & Eijkemans MJC. Selecting a classification function for class prediction with gene expression data. Bioinformatics (2016) 32(12): 1814-1822;

See Also

estimateDataCha, fitLMEModel and plotSPreFu

Examples

```
#Let us consider a single simulated train data as our real-life dataset
myCov<-covMat(pAll=100, lambda=2, corrDE=0.75, sigma=0.25);
myData<-generateGED(covAll=myCov, nTrain=30, nTest=10);
data<-myData[[1]]$trainData;
dataY<-myData[[1]]$trainLabels;
myDataCha<-estimateDataCha(data, dataY);
myFit<-fitLMEModel();  #Takes roughly 250 Sec
myPred<-SPreFu(myDataCha, myFit);
```

Index