Write-up for CTD-squared BeatAML DREAM Challenge

Wei Zhang

May 11, 2020

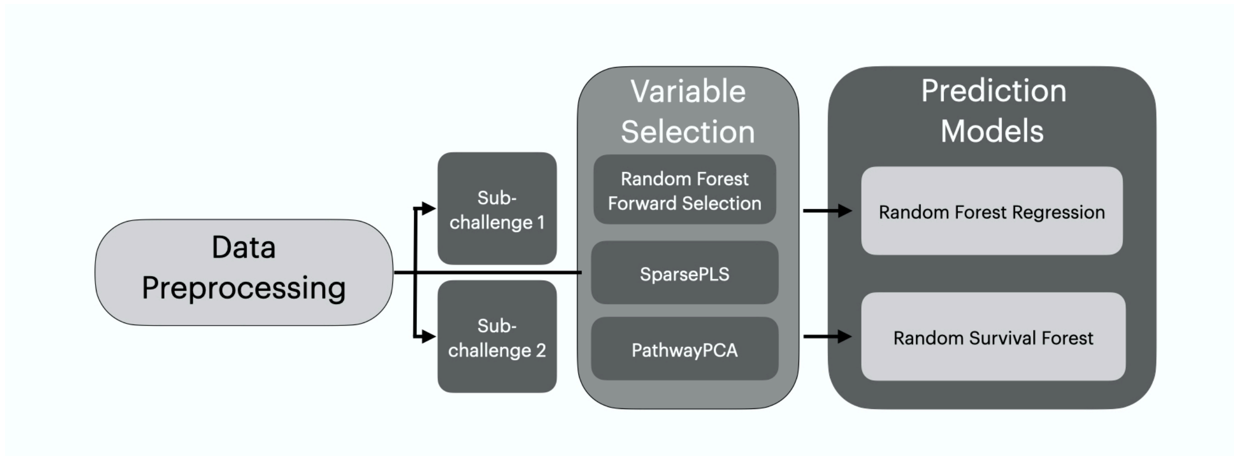
Introduction

The BeatAML DREAM challenge contains two sub-challenges:

- Predict the AUC values for 122 inhibitors. The training dataset contains clinical data, 63675 genes expression data, DNA sequence data and 122 auc values of 213 patients. In the leaderboard phase we need to predict 122 inhibitors auc values for 80 patients and combine the training data and leaderboard data to predict the validation set, which only contains 40 observations.
- Predict survival days after inclusion in the study. The sub-challenge contains all the datasets in sub-challenge 1 plus a survival response data in the training set and leaderboard phase.

The figure below shows the process of our works for the two sub-challenges (Figure 1). For both challenges, we implemented our models with R.

Figure 1: Pipeline of the challenge



Methods

1 Data Preprocessing

All datasets for both sub-challenges were preprocessing in the same way as shown in the introduction.

- RNA-expression data: We ordered the gene expression data with variance, and selected the first 1500 genes of the ordered gene expression. All genes were scaled and centered.
- Clinical data: Clinical data contains 19 categorical variables and 5 numerical variables. For the categorical variables, we transform them into 0-1 dummy variables. We only retained 1 numerical variables `ageAtDiagnosis` for further analysis, since other numerical variables contain too many missing values. We imputed the `ageAtDiagnosis` with a highly correlated variable named `ageAtSpecimenAcquisition`.

2 Variable Selection

Sub-challenge 1

In sub-challenge 1, we noticed that each drug might respond to different genes. With the same gene set, some drugs have high correlation responses, but some drugs have low responses or even get negative correlations. Here we extract the first 5 PCs of the genes for each drug as the plot shows in Figure 2. Therefore, gene selection is critical to predicting AUC scores.

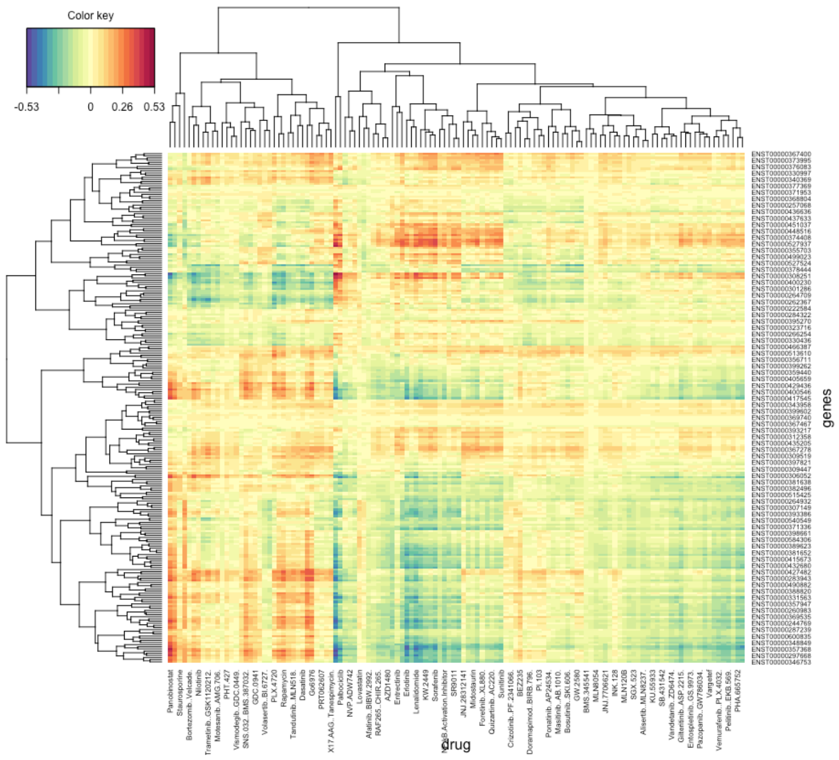
Random Forest Forward Selection

For sub-challenge 1, we implemented the RF forward selection method. We first run a full model that contained all the variables (clinical and gene) for each drug, then we selected the first 50 or 100 variables based on the RF variable importance. Since the result of sub-challenge 1 was evaluated by Spearman correlation, we added each variable in the 50 or 100 variables to the baseline model, which contained only selected clinical variables, and calculated the spearman correlation for each model. If the correlation for both training correlation and test correlation increase, we added this variable to our final selection. Here we seperated the training data into 0.85 training set and 0.15 validation set.
The implementation of the RF forward selection method is in the Appendix. The RF implementation was provided from the `randomForest` package in R.

Sub-challenge 2

We used different variable selection methods for sub-challenge 2 based on the challenge settings. In sub-challenge 2, we noticed that the clinical features performed much better result than the gene features. With the clinical features only, we got up to 0.70 on the C-index. However, adding the genes would lower the results Hence, we first performed the sparse partial least square method and

Figure 2: Heatmap of 5 components of genes and 122 drugs



pathwayPCA method to select the genes. Then we made a second selection based on the variable importance of the RSF.

SparsePLS

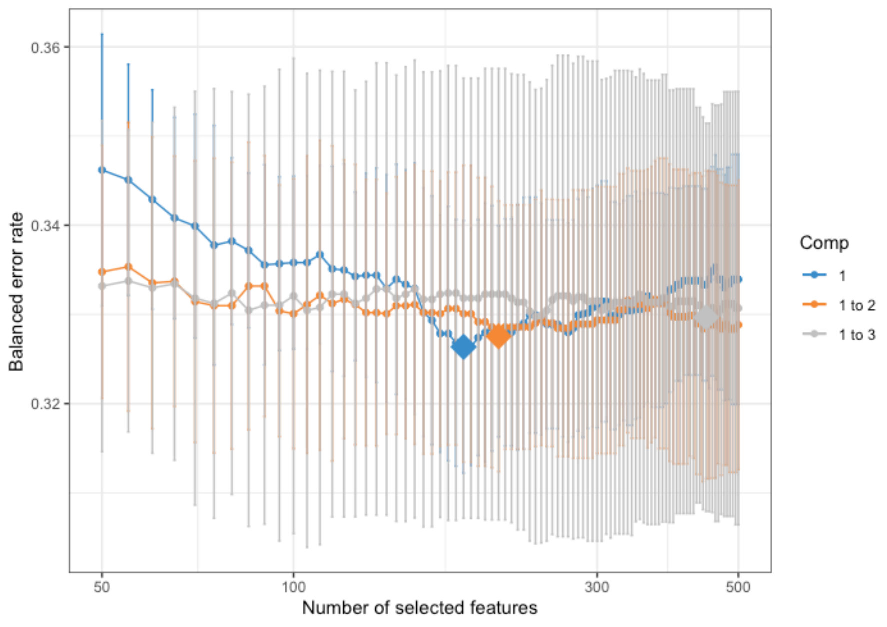
We first treated the survival response as the classification problem in SparsePLS. Based on the 5 folds cross-validation, we selected 185 genes with the 1st component, 210 genes with the 2nd component and 445 genes with the 3rd component. Figure 3 shows the error rate of different selections with 3 comps.
The sparsePLS implementation was provided from the `mixOmics` package in R.

pathwayPCA

`PathwayPCA` is an R package that combines pathway analysis with PCA and gene selection. It contains two PCA methods: Supervised PCA (SuperPCA) and Adaptive, elastic-net, sparse PCA (AES-PCA), to extract relevant genes from pathways. In sub-challenge 2, we use the AES-PCA approaches to extracting the relevant genes since it calculates the p-values non-parametrically and it can handle missing values in response. The pathways collection we used was the MsigDB C2-CP collection.

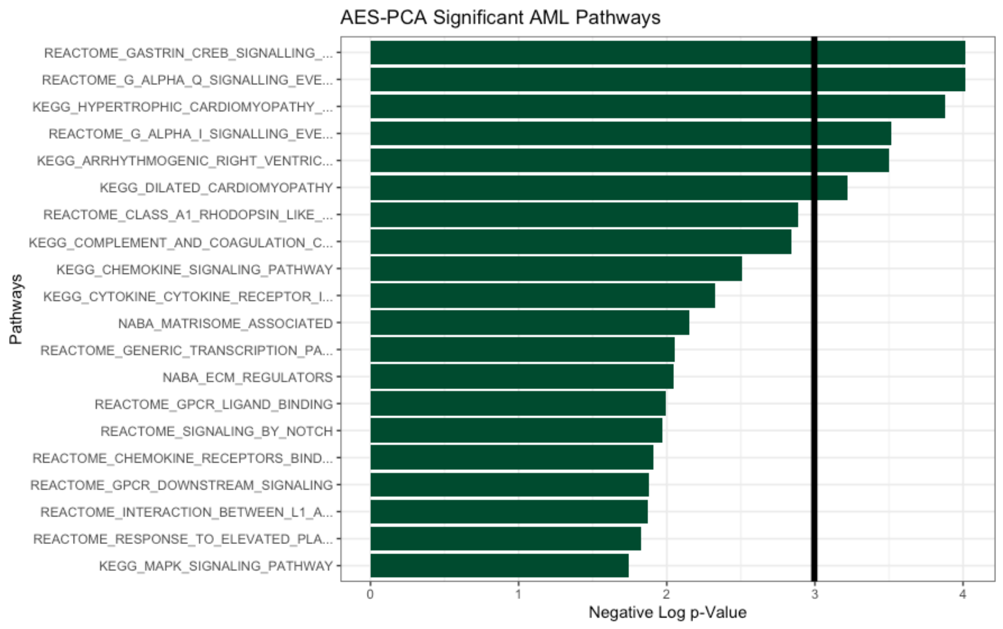
Figure 4 shows the negative log raw p-values for the pathways selected by pathwayPCA. The black line represents the threshold $-\log(0.05)$. We extracted the relevant genes in the selected pathways

Figure 3: Error rate of selecting different number of features



for further model predictions.

Figure 4: P-values for 20 pathways



3 Prediction Models

Random Forest Regression

The RF regression implementation was provided from the `randomForest` package in R. In the sub-challenge 1, we set 1500 trees for each model and applied the selected variables to predict AUC values for each drug.

Here are the results of some of the models we perform for the leaderboard data (Table 1), the methods are tested in local:

Table 1: Results of sub-challenge 1

Methods for sub-challenge 1	**Mean Sp Corr**
1. No selection with random choose of genes + RF	0.26-0.27
2. No selection with most 1500 variate genes + clinical features + RF	0.27-0.28
3. RF forward selection with random choose of genes + clinical features +RF	0.36-0.40
4. RF forward selection with most 1500 variate genes + clinical features +RF	**0.52-0.58**

Random Survival Forest

Since we were predicting the survival days or inverse harzard for sub-challenge 2, we performed the Random Survival Forest (RSF) as our prediction model. RSF is an ensemble tree-based method for analyzing the right-censored survival data. The cumulative hazard function (CHF) is obtained by averaging the CHF for each tree. We set 0 = right censored (Alive) and 1 = Death, and the splitting rule was the logrank splitting. The risk was predicted by the model with the variables selected in the previous stage.

The implementation of RSF was provided from the `randomForestSRC` package in R. Here are the results of some of the models we perform for the leaderboard data and randomly separated test data (Table 2). The methods were tested in local:

Table 2: Results of sub-challenge 2

Methods for sub-challenge 2	**C-index**
1. Selection with clinical features only + RSF	0.69-0.71
2. No selection with gene features only + RSF	0.43-0.53
3. SparsePLS selection with genes + clinical features + RSF	0.56-0.66
4. PathwayPCA selection with genes + clinical features + RSF	**0.67-0.74**

Discussion

Although the RF forward selection method performed high Spearman correlation scores, it suffered from the randomness of gene selection, in other words, it selected different sets of genes each time performed. Therefore, we combined the gene sets with different performance in the final submission. In sub-challenge 2, the best result we had for leaderboard data was 0.76, but the results were varied from different validation sets with the same set of features. For final submission, we selected the RF forward selection method (4) for sub-challenge 1 and pathwayPCA selection method (4) for the sub-challenge 2.

Reference

1. CTD-squared BeatAML DREAM Challenge (syn20940518)
2. A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. R News 2(3), 18–22.
3. Odom, Gabriel J., et al. PathwayPCA: an R Package for Integrative Pathway Analysis with Modern PCA Methodology and Gene Selection. 2019, doi:10.1101/615435.
4. Chen, Xi. Adaptive Elastic-Net Sparse Principal Component Analysis for Pathway Association Testing. Statistical Applications in Genetics and Molecular Biology, vol. 10, no. 1, 2011, doi:10.2202/1544-6115.1697.
5. Ishwaran H, Kogalur U (2007). Random survival forests for R. R News, 7(2), 2531. https://cran.r-project.org/doc/Rnews/.

Appendix

```
best_genes = function(train_sample,test_rna){
     gene.list = NULL
     gene = NULL
     set.seed(510)
     rf.mod=randomForest(auc~.,importance = T,
                              data = train_sample,ntree = 1500,nodesize = 3)
     p1 = predict(rf.mod,OOB = T)
     cor.train = cor(p1,\begin{commandline}
\begin{verbatim}train_sample$auc,method = "spearman")
     p2 = predict(rf.mod,test_rna,OOB=T)
     cor.test = cor(p2,test_rna$auc,method = "spearman")
     im_gene = paste(names(sort(importance(rf.mod,type = 1)[,1],
     decreasing =T)[1:100]))

     clinical_names = "ageAtDiagnosis+FLT3.ITD1+priorMDS1+NPM11+
     specificDxAtAcquisition12+dxAtSpecimenAcquisition2"
     response = "auc"

     for (i in c(1:100)){
        temp = im_gene[i]
        if(length(gene.list) == 0){
           modelformula = paste(response,"~",clinical_names,"+",temp)
         }else{
           modelformula = paste(response,"~",clinical_names,"+",temp,gene.list)
         }
         rf.new = randomForest(as.formula(modelformula),data = train_sample,
         ntree = 1500)
         p1 = predict(rf.new,OOB = T)
         cor.train.update = cor(p1,train_sample$auc,method = "spearman")
         p2 = predict(rf.new,test_rna,OOB=T)
         cor.test.update = cor(p2,test_rna$auc,method = "spearman")
         a = cor.train.update - cor.train
         b = cor.test.update - cor.test
```

```
   if(a>-0.1&b>0){
      gene.list = paste(gene.list,"+",temp)
      gene= c(gene,temp)
    }else{
      gene.list= gene.list
      gene = gene
    }
}
```