CYO Project Capstone Edx

Hildsley Noome

11 February 2019

Introduction

The human body is extraordinary in the sense that it exists out of billions of cells, where each cell maintains a balance with thousands of different pathways. The cells are able to communicate with each other even though they could be in completely different parts of the body. This complex interaction and pathways should be maintained while the cells grow[2].

Cells go through a cycle commonly referred to as mitosis. This cycle is important to ensure that cells can grow and replicate. During mitosis, it is crucial that the DNA is replicated for subsequent division of a single cell into two completely new cells. Often the replication pathways make errors in copying the DNA, but redundancy systems in the cell more often than not catch the error and correct it. If the error is missed, the new cell will carry this error in its DNA – known as DNA mutations or damage[1].

DNA mutations or damage may be in a crucial area of the DNA which is responsible for growth and maintenance of cells. In this case, the damage would lead to uncontrolled growth of cells and a tumour may develop. Tumours that continue to grow uncontrollably and spread to different parts of the body are commonly referred to as cancer[3,4]. Cancer causes approximately 8.5 million deaths per year[5].

There are several intervention types to combat cancer. A common intervention is to surgically remove the tumour to prevent further growth and/or spreading and could lead to a better prognosis. Therefore any data related to cancer and tumour growth may be valuable and informative for diagnostic and prognosis purposes.

This collection of data is part of the RNA-Seq (HiSeq) PANCAN data set, it is a random extraction of gene expressions of patients having different types of tumours. Samples (instances of different patients) are stored row-wise. Variables (attributes) of each sample are RNA-Seq gene expression levels measured by an Illumina HiSeq platform[7,8].

The cancer types in the dataset[6]:

BRCA = Breast Invasive Carcinoma

COAD = Colon Adenocarcinoma

KIRC = Kidney Renal Clear Cell Carcinoma

LUAD = Lung Adenocarcinoma

PRAD = Prostate Adenocarcinoma

The dataset contains 801 samples of patients and a total of 20533 columns.

The goal of this project is to build a Machine Learning algorithm which can identify which cancer type the gene expression levels of the unknown sample belongs to.

The main steps in this report include dimension reduction, which is necessary due to the large number genes, and training a model to a training subset from which the model will learn and build a prediction algorithm to then predict which cancer type the unknown sample belongs to.

Analysis

Creating Datasets

The Large dataset was divided into two subsets, a training and validation dataset. The training dataset will be used to build a model and the validation dataset will be used to test how accurately the model can predict the cancer type from the genes' relative expression levels.

The Following code creates these two subsets:

```
set.seed(2019) # set a random number to ensure reproducibility of the data
contained in the different datasets
ind <- createDataPartition(y = rna_seq_dat$Class, times = 1,p = 0.2, list =
FALSE)
# creates the indexes for creating the training
# and validation datasets, 20 % of the original dataset
# will be allocated towards the validation set of samples.

rna_seq_val <- rna_seq_dat[ind,]  # Splices out the validation samples from
the original dataset
rna_seq_train <- rna_seq_dat[-ind,] # Splices out the training samples for ML
from the original dataset
```

The following piece of code ensure no samples have been lost during the data partitioning step:

```
nrow(rna_seq_val) + nrow(rna_seq_train) == nrow(rna_seq_dat)

## [1] TRUE
```

Evaluating Training Dataset

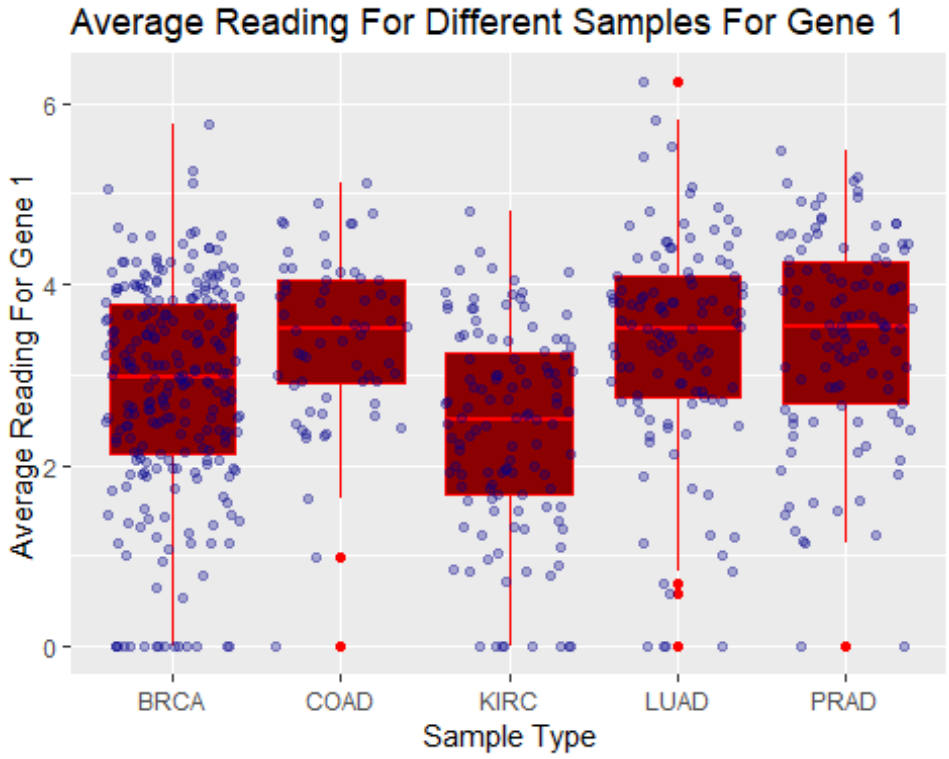
The training set was investigated to determine exactly how the data looks like and what kind of techniques could be of use to build a machine learning model. The Following code shows the first few samples and genes' expression levels:

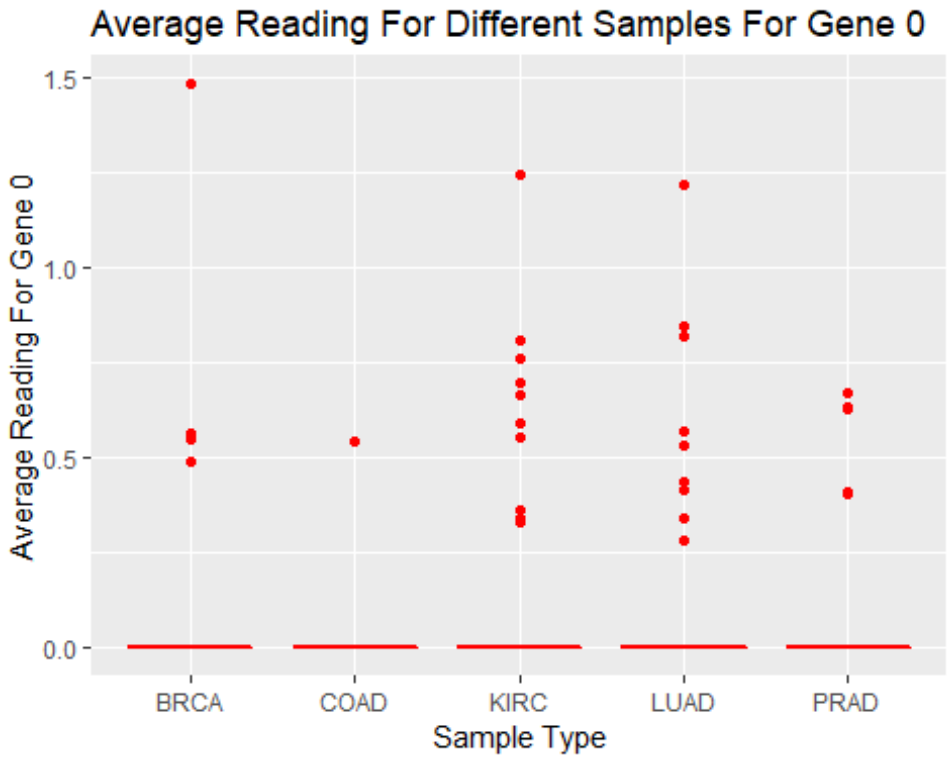
```
rna_seq_train[1:6,1:6] %>% knitr::kable()
```

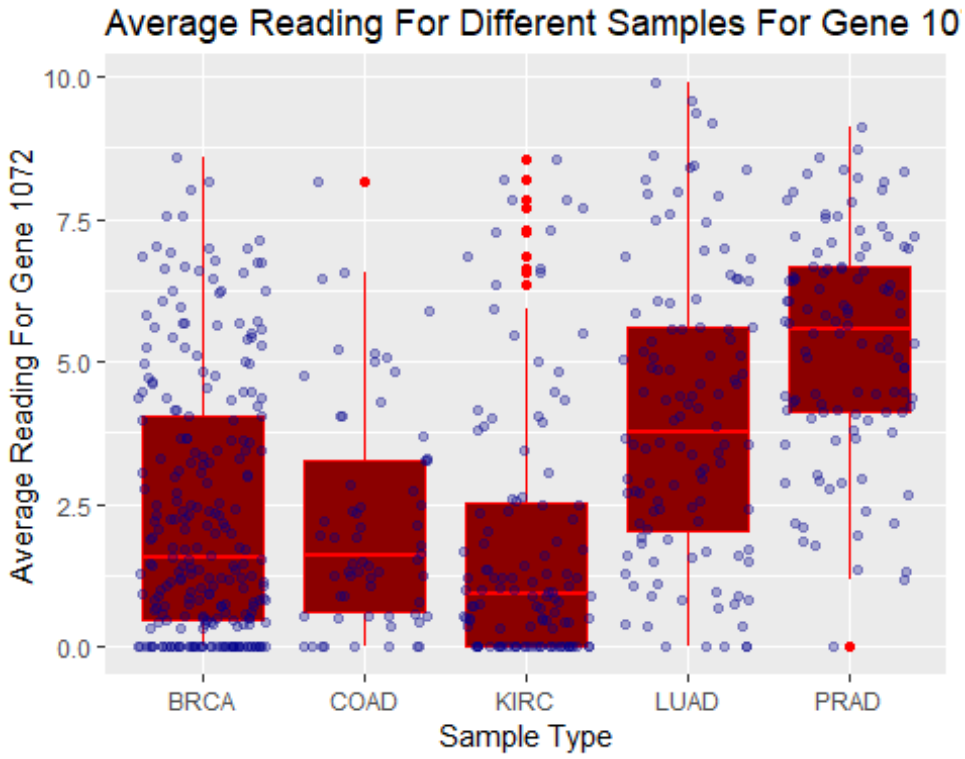
	sample_number	Class	gene_0	gene_1	gene_2	gene_3
2	sample_1	LUAD	0	0.5927321	1.588421	7.586157
3	sample_2	PRAD	0	3.5117590	4.327199	6.881787
4	sample_3	PRAD	0	3.6636179	4.507649	6.659068
5	sample_4	BRCA	0	2.6557411	2.821547	6.539453
6	sample_5	PRAD	0	3.4678533	3.581918	6.620243
7	sample_6	KIRC	0	1.2249664	1.691177	6.572007

From the table, it is clear that we have different types of cancer and then the gene expression levels in the following columns, which contains numeric values. The numeric values are of importance, especially due to the fact that dimensionality reduction methods could prove to be very important for this kind of data.

The following graphs show how different genes' expression levels vary significantly. This is important as many genes' expression levels may not inherently possess predictive power suitable for building a machine learning algorithm.







The graphs show how the different genes' expression levels may have different degrees of predictive power. Therefore the genes that do not inherently have any predictive power should be removed.

Principal Component Analysis

Principal component analysis (PCA) is a very powerful dimensionality reduction technique. This technique possibly reduces the dimensions of a dataset by including those components that add variance to the data. Therefore dimensions that do not add variance to the data could be removed as they inherently do not contain any information. In the training dataset and the graphs above, it was shown that many genes' expression levels do not have much predictive power. Therefore it is expected that a PCA would reduce the dimensions significantly.

The following code performs a PCA on the training dataset:

```
pca_rna_seq_train <- prcomp(x = rna_seq_train[,3:ncol(rna_seq_train)], center
= TRUE) # Performs a PCA analysis on all the genes present in the dataset

str(pca_rna_seq_train) # Gives an indication of the structure and values
after the pca analysis

## List of 5
##  $ sdev    : num [1:638] 75.3 61.9 59 48.6 36.9 ...
##  $ rotation: num [1:20531, 1:638] 0.000178 -0.002991 -0.003749 -0.001799 -
0.002764 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:20531] "gene_0" "gene_1" "gene_2" "gene_3" ...
##   .. ..$ : chr [1:638] "PC1" "PC2" "PC3" "PC4" ...
##  $ center  : Named num [1:20531] 0.029 2.983 3.087 6.708 9.815 ...
##   ..- attr(*, "names")= chr [1:20531] "gene_0" "gene_1" "gene_2" "gene_3"
...
##  $ scale   : logi FALSE
##  $ x       : num [1:638, 1:638] -1.95 -70.63 -83.59 -69.43 -35.28 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:638] "2" "3" "4" "5" ...
##   .. ..$ : chr [1:638] "PC1" "PC2" "PC3" "PC4" ...
##  - attr(*, "class")= chr "prcomp"

dim(pca_rna_seq_train$rotation) # Dimensions shows the amount of Principal
Components, 20531 genes and 638 principal components

## [1] 20531   638

pca_rna_seq_train$x[1:5,1:7] # Shows the first few PCs

##          PC1       PC2       PC3         PC4        PC5        PC6
## 2  -1.946371 -86.48328 -18.92916  55.7430779 -14.943455  58.782355
## 3 -70.632195  -9.50537  68.92108 -79.1049004  -7.653349 111.387170
## 4 -83.588072  56.38757  87.70425 -27.8992551 -19.899707  52.187084
## 5 -69.431686  21.88504 -63.72219 -36.3067667  -5.088313  -6.803359
## 6 -35.280128  90.28995  72.08289  -0.1141885   2.292501  -6.623736
##           PC7
## 2 -21.3947447
```

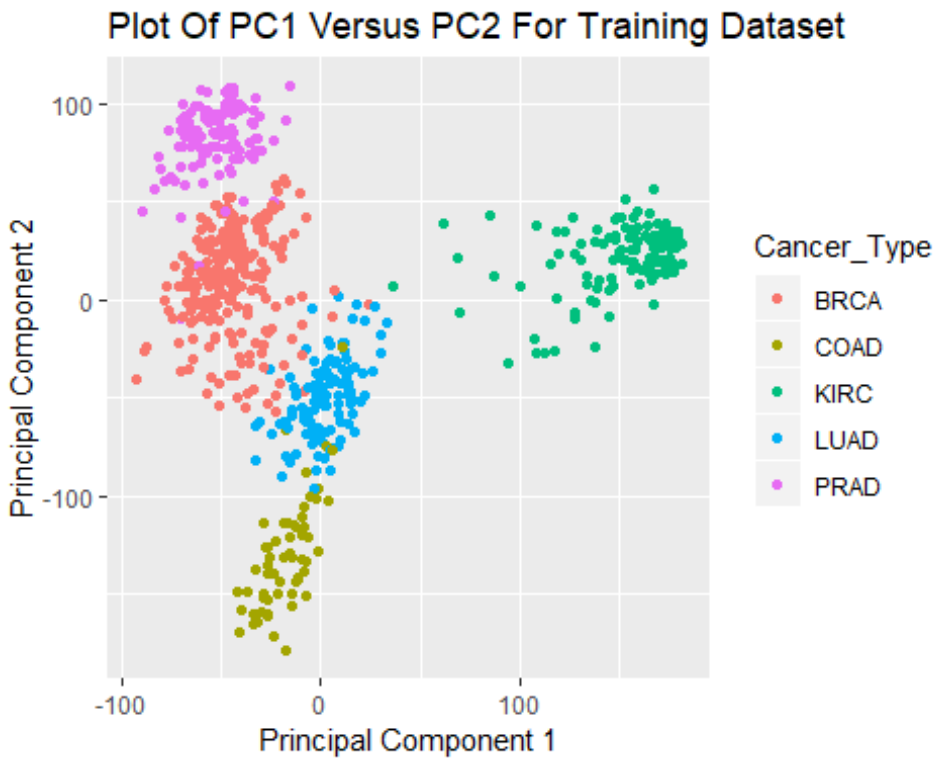
```
## 3 -16.4484204
## 4   0.3155161
## 5   5.2942179
## 6 -18.8316761
```

The PCA reduced the predictors from an initial 20531 to 638. This is a much better size of predictors to work with and build a model.

The following graph shows principal component 1 (PC1) and PC2 for the training dataset:

```
pca_ggplot <- data.frame(rna_seq_train[,2] , pca_rna_seq_train$x)
colnames(pca_ggplot)[1] <- "Cancer_Type"
pca_ggplot[,1] <- as.factor(pca_ggplot[,1])

pca_ggplot %>% ggplot(aes(x = PC1, y = PC2, color = Cancer_Type)) +
  geom_point() +
  ggtitle(label = "Plot Of PC1 Versus PC2 For Training Dataset" ) +
  xlab(label ="Principal Component 1") +
  ylab(label = "Principal Component 2")  # Plot showing PC1 and PC2 of
classes with good clustering
```



The PCA graph shows how the principal components could be used to explain the variance and identifies clustering of the different cancer types. This graph shows that following a PCA, a model could be trained to predict cancer types given the different principal components.

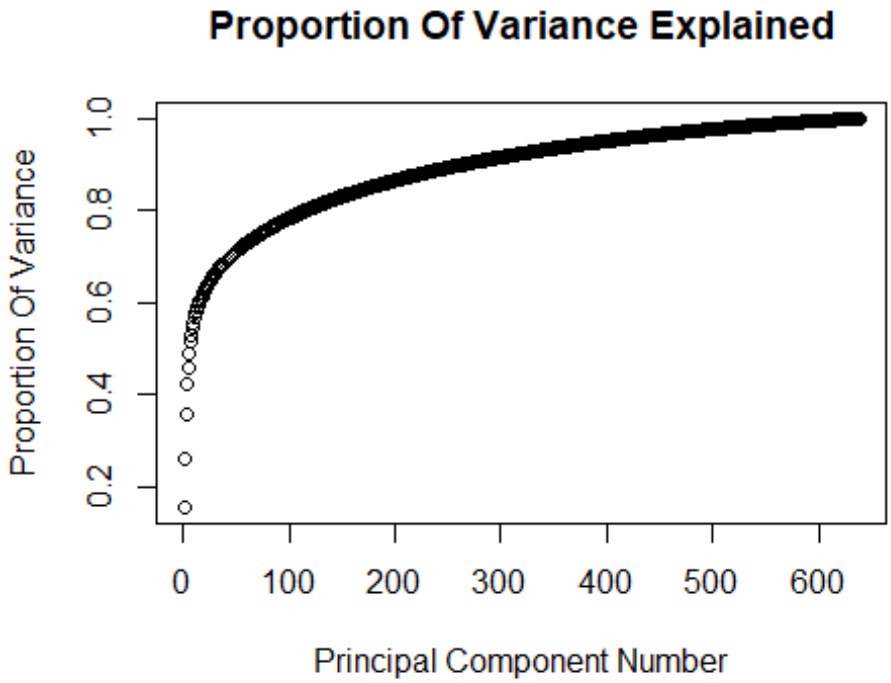
Dimensionality Reduction

Following the PCA, it was shown that the number of predictors reduced significantly. The number of dimensions could be reduced even further by looking at how much variance each principal component explains of the original dataset.

The following plot shows the cumulative variance that is explained by the different principal components.

```
pca_var <- pca_rna_seq_train$sdev^2 # Computes the variance of each PC
prop_var <- pca_var/sum(pca_var) # calculates the proportion each PC adds to
the total variance

plot(x = 1:length(prop_var), y = cumsum(prop_var), main = "Proportion Of
Variance Explained",
                                                    xlab = "Principal Component
Number",
                                                    ylab = "Proportion Of
Variance" ) # Plot showing how the different Principal components add to the
total variance
```



The plot shows that the amount of predictors that inherently explain 90% of the variance are 263 and 80% variance are 115.

The cutoff value of the total variance included by the PC's should be carefully decided. We would like to include as much of the predictors as possible while being efficient in using

computer processing power. For 90% of the variance explained we further reduced the predictors by approx. a half (0.4122257) while for 80% of the variance explained the predictors reduced to approx. a fifth (0.1802508). Following these findings, both the 90% principal components and 80% principal components will be used to comparatively train a model.

Training The Model

Following the PCA, different models will be trained given the training dataset with the 80% and 90% variance datasets. The training will be done with 80% cross-validation sampling of the training set a total of 10 times.

The following code will perform the training of 10 different models and print out the accuracies for the 80% variance set:

```
pca_rna_seq_train_80 <- data.frame(
rna_seq_train[,2],pca_rna_seq_train$x[,1:min(which(cumsum(prop_var)> 0.8))])
# Joins the classes of the samples with the respective
# PCs after the pca, for the 80% variance PCs.

colnames(pca_rna_seq_train_80)[1] <- "sample_class" # change column name to
sample class

models <- c("bayesglm",
"rpart","knn","svmLinear3","lda","naive_bayes","pls","snn","svmLinear","svmRa
dial") # Different models which will be trained

fitControl <- trainControl(method = "cv", number = 10 , p = 0.8) # Control
ensuring a cross-validation of 10 times would be completed on the training
set of 80%


fits_80 <- lapply(models, function(models){
  set.seed(2020)
  train(sample_class ~ . , data = pca_rna_seq_train_80, method = models ,
trControl = fitControl)
})

method_80 <- matrix() # variable to store the methods
acc_80 <- matrix() # Variable to store the accuracies

for (x in 1:length(fits_80)) {
  method_80[x] <- (fits_80[[x]][1]$method)
  acc_80[x] <- (max(fits_80[[x]][4]$results$Accuracy))
}
acc_list_80 <- data.frame(method_80,acc_80)

acc_list_80 %>% knitr::kable() # Shows the accuracies for each model
```

method_80	acc_80
bayesglm	0.4718463
rpart	0.9076002
knn	0.9967742
svmLinear3	1.0000000
lda	1.0000000
naive_bayes	0.9733863
pls	0.9012246
snn	0.4734088
svmLinear	0.9984127
svmRadial	0.9701102

The following code will perform the training of 10 different models and report the accuracies for the 90% variance set:

```
pca_rna_seq_train_90 <- data.frame(
rna_seq_train[,2],pca_rna_seq_train$x[,1:min(which(cumsum(prop_var) > 0.9))])
# same joining as above for the 90% variance PCs

colnames(pca_rna_seq_train_90)[1] <- "sample_class"

fits_90 <- lapply(models, function(models){
  set.seed(2020)
  train(sample_class ~ . , data = pca_rna_seq_train_90, method = models ,
trControl = fitControl)
})


method_90 <- matrix()
acc_90 <- matrix()

for (x in 1:length(fits_90)) {
  method_90[x] <- (fits_90[[x]][1]$method)
  acc_90[x] <- (max(fits_90[[x]][4]$results$Accuracy))
  }
acc_list_90 <- data.frame(method_90,acc_90)

acc_list_90 %>% knitr::kable() # Shows the accuracies for each model
```

method_90	acc_90
bayesglm	0.4623961
rpart	0.9076002
knn	0.9983871
svmLinear3	1.0000000

lda	0.9968246
naive_bayes	0.9483809
pls	0.9012246
snn	0.4734088
svmLinear	0.9826861
svmRadial	0.8290666

The 80% variance set's models had similar accuracies to the 90% variance set, in fact, 5 models had higher accuracies for the 80% variance set. The accuracies between the 80% and 90% variance sets did not give any indication why the 90% variance dataset that contains more predictors should be considered above the 80% variance training set. The 90% variance dataset cumsumes more processing power as well and therefore the 80% variance training models would be used for evaluating the accuracy on the validation set.

Best Model

The svmLinear 3 (L2 regularized support vector machine with the linear kernel) and LDA (linear discriminant analysis) models were perfectly accurate after training on the 80% variance training set. Choosing the best model would now depend on which model's computation time is the least. This may be very important when this dataset includes more training samples to build a better and expanded machine learning algorithm. For example, if the modelling dataset in the future now expanded to include other cancer types as well, together with more patients' gene expression levels, this difference may become a very important factor to consider.

The following code evaluates the time it takes training these models:

```
time_svmL <- benchmark(train(sample_class ~ . , data = pca_rna_seq_train_80 , method = "svmLinear3", trControl = fitControl , tuneGrid = expand.grid(cost = 0.25, Loss = 1)),
                       columns = c("elapsed"), replications = 10) # runs the model fitting algorithm 10 times and records the time it takes to complete.

# The expand grid best tune variables was used as was determined in the original fits_80 piece of code
# fits_80[[4]][6]

time_lda <- benchmark(train(sample_class ~ . , data = pca_rna_seq_train_80, method = "lda" , trControl = fitControl), columns = c("elapsed"),replications = 10 )

times <- data.frame(time_svmL) %>% rbind(time_lda)#
rownames(times) <- c("svmL","lda")                # Shows that the computation time for lda is better
times                                             #
```

```
##      elapsed
## svmL   21.35
## lda     9.94
```

Following the computation time for each model, it was shown that LDA was more than twice as fast than svmLinear 3. The LDA model would then be used as the best model to predict the cancer types of samples that is unknown.

Results

The LDA model that was trained was tested on a validation set which was completely separate from the training set during the whole modelling phase.

The following code trains the LDA model on the training set:

```
fit_lda <- train(sample_class ~ . , data = pca_rna_seq_train_80, method =
"lda",trControl = fitControl)
```

The validation set that was separate the whole modelling phase will now be prepared for validating the LDA model. The validation set was not included in the PCA and therefore the values of each samples' principal components are unknown. The principal components of these samples will be fitted unto the PCA of the training set. It is important to notice that the original principal components do not change nor do we perform a PCA on the validation set. The validation set is being transformed to the same rotations and components of the original PCA so that we know where do these validation points lie in terms of the PCA that was already performed relative to those principal components of the training dataset.

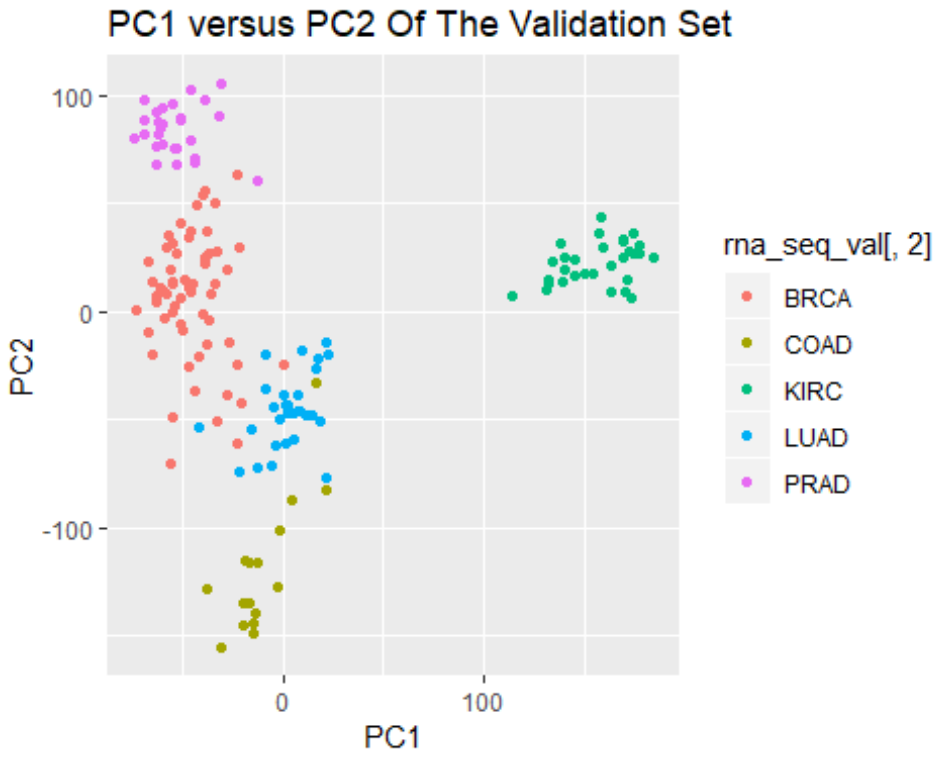
The following code performns the transformation of the validation set:

```
pca_rna_seq_val <- predict(pca_rna_seq_train,newdata = rna_seq_val)
#Transform validation set into similar pca parameters of the training set

pca_rna_seq_val <- data.frame(pca_rna_seq_val[,1:min(which(cumsum(prop_var)>
0.8))] ) # removes the PCs we are not interested in and adds the sample's
classes to the data frame
```

The following graph shows the principal component values of the validation set, relative to the training set's PCA.

```
pca_rna_seq_val %>% ggplot(aes(x=PC1, y = PC2, color = rna_seq_val[,2])) +
  geom_point() +
  ggtitle(label = "PC1 versus PC2 Of The Validation Set") +
  xlab(label = "PC1") +
  ylab(label = "PC2")
```



This graph closely resembles that of the PC1 and PC2 of the training set graph. From this graph, one can easily see the clustering that is as expected following the PCA of the training set. The model will now be predicting the cancer types given the principal components of the validation set relative to the PCA of the training set.

The following code performs the predictions and then prints out a confusion matrix:

```
pred_val <- predict(fit_lda,pca_rna_seq_val) # Predicts the classes of
validation set given the model that is fitted to the training dataset

confusionMatrix(pred_val,rna_seq_val[,2]) # Confusion matrix showing how well
the model performs

## Confusion Matrix and Statistics
##
##           Reference
## Prediction BRCA COAD KIRC LUAD PRAD
##       BRCA   60    0    0    0    0
##       COAD    0   16    0    0    0
##       KIRC    0    0   30    0    0
##       LUAD    0    0    0   29    0
##       PRAD    0    0    0    0   28
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9776, 1)
```

```
##     No Information Rate : 0.3681
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: BRCA Class: COAD Class: KIRC Class: LUAD
## Sensitivity               1.0000     1.00000       1.000      1.0000
## Specificity               1.0000     1.00000       1.000      1.0000
## Pos Pred Value            1.0000     1.00000       1.000      1.0000
## Neg Pred Value            1.0000     1.00000       1.000      1.0000
## Prevalence                0.3681     0.09816       0.184      0.1779
## Detection Rate            0.3681     0.09816       0.184      0.1779
## Detection Prevalence      0.3681     0.09816       0.184      0.1779
## Balanced Accuracy         1.0000     1.00000       1.000      1.0000
##                      Class: PRAD
## Sensitivity               1.0000
## Specificity               1.0000
## Pos Pred Value            1.0000
## Neg Pred Value            1.0000
## Prevalence                0.1718
## Detection Rate            0.1718
## Detection Prevalence      0.1718
## Balanced Accuracy         1.0000
```

Following the analysis of the predictions, the model perfectly predicted the cancer types of the validation step.

Conclusion

The LDA model perfectly predicted each cancer type from the validation step. This result is excellent since detection of cancer types may be very valuable, especially if one would like to allocate risk factors to different patients. One shortcoming though is that the information regarding the cancer types is quite limited. There was no specification of the stage these patients were at the time of the RNA-sequencing.

The results confirm that it is possible to predict a cancer type from genes' expression levels and a physician may make an earlier diagnostic decision regarding a specific patient without requiring more invasive methods.

References

1. "The cell cycle: a review of regulation, deregulation and therapeutic targets in cancer", *Cell Prolif. 2003, 36, 131–149*, Katrien Vermeulen, Dirk R. Van Bockstaele and Zwi N. Berneman, Faculty of Medicine, Laboratory of Experimental Hematology, University of Antwerp, Antwerp University Hospital,Edegem, Belgium

2. "The biology of cancer stem cells", *Annu. Rev. Cell Dev. Biol. 2007. 23:675–99*, Neethan A. Lobo[12], Dalong Qian[2], Yohei Shimono[2] and Michael F. Clarke[2] [1]Cellular and Molecular Biology Program, University of Michigan, Ann Arbor, Michigan 48109; email: nlobo@umich.edu [2]Institute for Stem Cell Biology and Regenerative Medicine, Stanford University,Palo Alto, California 94304; email: yshimono@stanford.edu, dqian@stanford.edu, mfclarke@stanford.edu

3. "Cancer", *World Health Organization*, retrieved 13 February 2019, https://www.who.int/en/news-room/fact-sheets/detail/cancer.

4. "What is Cancer", *National Cancer Institute*, retrieved 13 February 2019, https://www.cancer.gov/about-cancer/understanding/what-is-cancer.

5. "Global, regional, and national life expectancy, all-cause mortality, and cause-specific mortality for 249 causes of death, 1980–2015: a systematic analysis for the Global Burden of Disease Study 2015", *Lancet* , 2016 Oct 8.

6. "List of Cancer-types", *MAGI: Mutation Annotation & Genome Interpretation*, retreived 13 February 2019, http://magi.brown.edu/cancers.

7. "The Cancer Genome Atlas Pan-Cancer analysis project", *Nat Genet. 2013 Oct;45(10)*, Cancer Genome Atlas Research Network1, Weinstein JN, Collisson EA, Mills GB, Shaw KR, Ozenberger BA, Ellrott K, Shmulevich I, Sander C, Stuart JM.

8. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml], Dua, D. and Karra Taniskidou, E. (2017), Irvine, CA: University of California, School of Information and Computer Science.