Final Project

Feilong Hou

2021/3/14

Introduction

This is the final project for MATH 189.
In this project, we need to exam the Swiss Bank Note dataset(SBN.txt) in order to predict whether a note is false or counterfeit using supervised learning.

Inport The Dataset

```
data = read.table("C:/Users/h1206/Desktop/MATH 189/Data/SBN.txt", header=TRUE)
head(data)
```

```
##     Length  Left Right Bottom  Top Diagonal
## BN1  214.8 131.0 131.1    9.0  9.7    141.0
## BN2  214.6 129.7 129.7    8.1  9.5    141.7
## BN3  214.8 129.7 129.7    8.7  9.6    142.2
## BN4  214.8 129.7 129.6    7.5 10.4    142.0
## BN5  215.0 129.6 129.7   10.4  7.7    141.8
## BN6  215.7 130.8 130.5    9.0 10.1    141.4
```

```
str(data)
```

```
## 'data.frame':    200 obs. of  6 variables:
##  $ Length  : num  215 215 215 215 215 ...
##  $ Left    : num  131 130 130 130 130 ...
##  $ Right   : num  131 130 130 130 130 ...
##  $ Bottom  : num  9 8.1 8.7 7.5 10.4 9 7.9 7.2 8.2 9.2 ...
##  $ Top     : num  9.7 9.5 9.6 10.4 7.7 10.1 9.6 10.7 11 10 ...
##  $ Diagonal: num  141 142 142 142 142 ...
```

```
library(ISLR)
library(logistf)
library(lattice)
library(ellipse)
```

```
##
## Attaching package: 'ellipse'
```

```
## The following object is masked from 'package:graphics':
##
##     pairs
```

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
library(e1071)
```

- This is a dataset that contains six variables measured on 100 genuine and 100 counterfeit old Swiss 1000-franc bank notes.
- Six variables are measured:

1. Length of the note
2. Width of the Left-Hand side of the note
3. Width of the Right-Hand side of the note
4. Width of the Bottom Margin
5. Width of the Top Margin
6. Diagonal Length of Printed Area

NOTE: BN1 to BN100 are genuine banknotes and BN101 to BN200 are counterfeit banknotes.
And each bank note is independent from others.

Source: The data is from "Multivariate Statistics: A practical approach", by Bernhard Flury and Hans Riedwyl, Chapman and Hall, 1988.

Analysis

Task 1: Explore and visualize the data.

Since we know the first 100 observations are the real banknotes and the rest are counterfeit. We need to label them first.

```
label = c(rep(0,100),rep(1,100)) # 0 is real, 1 is fake
swiss = cbind(data, label)
head(swiss)
```

```
##     Length  Left Right Bottom  Top Diagonal label
## BN1  214.8 131.0 131.1    9.0  9.7    141.0     0
```

```
## BN2  214.6 129.7 129.7    8.1  9.5    141.7     0
## BN3  214.8 129.7 129.7    8.7  9.6    142.2     0
## BN4  214.8 129.7 129.6    7.5 10.4    142.0     0
## BN5  215.0 129.6 129.7   10.4  7.7    141.8     0
## BN6  215.7 130.8 130.5    9.0 10.1    141.4     0
```

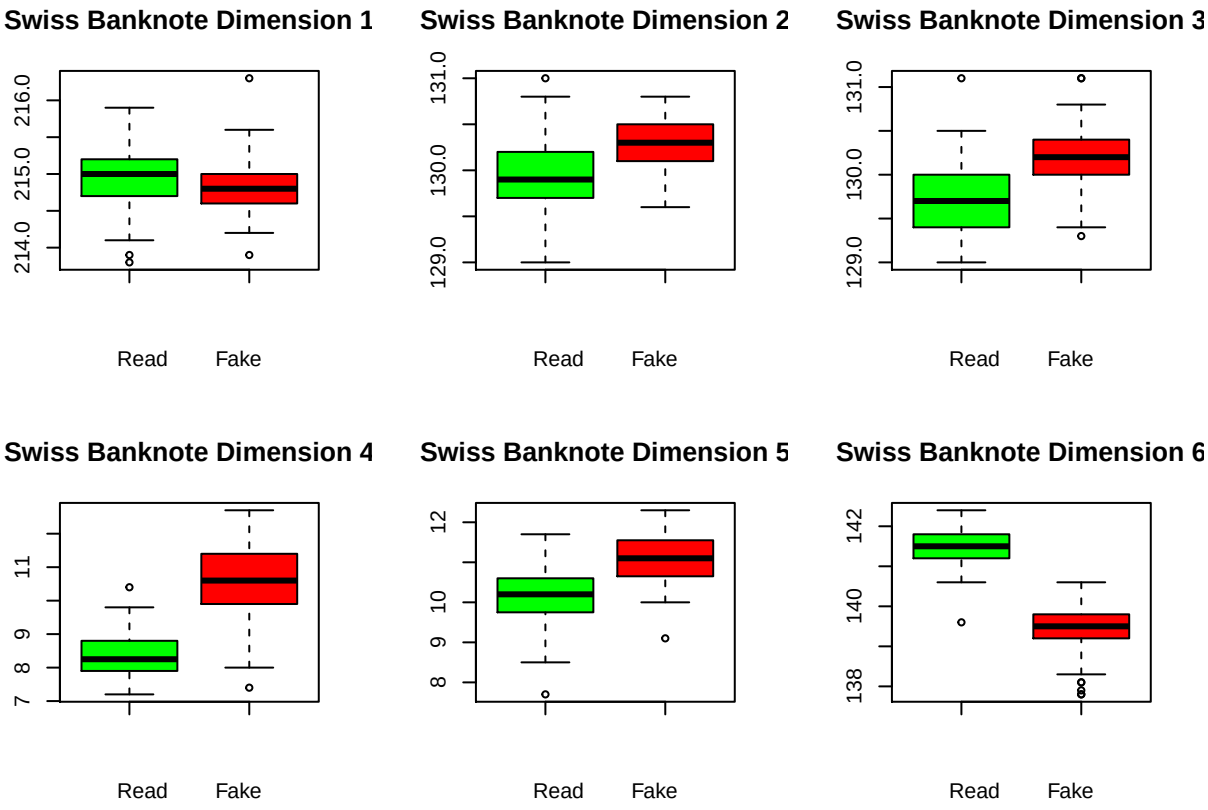
Now we need to compare sample means to see if there is a difference between real and counterfeit:

```
mean_mat = cbind(colMeans(swiss[1:100,]), colMeans(swiss[101:200,]))
colnames(mean_mat) = c("Genuine Sample Mean ","Counterfeit Sample Mean")
mean_mat
```

```
##          Genuine Sample Mean  Counterfeit Sample Mean
## Length               214.969                 214.823
## Left                 129.943                 130.300
## Right                129.720                 130.193
## Bottom                 8.305                  10.530
## Top                   10.168                  11.133
## Diagonal             141.517                 139.450
## label                  0.000                   1.000
```

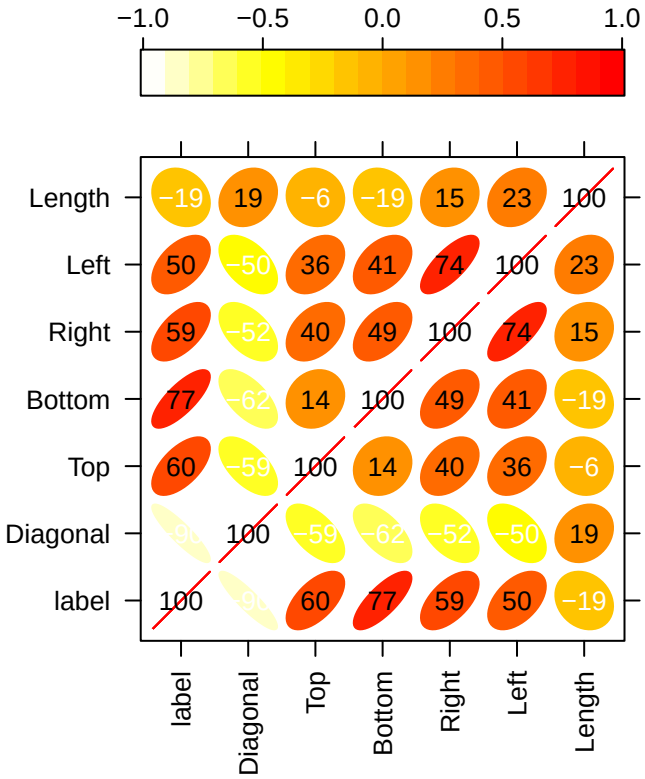
Plot them to visualize:

```
swiss_dim = dim(data)[2]
swiss_real = swiss[1:100,]
swiss_fake = swiss[101:200,]
par(mfrow = c(2,3))

for (i in 1:swiss_dim)
{
  title = paste0("Swiss Banknote Dimension ", i)
  boxplot(swiss_real[,i], swiss_fake[,i], col = c("green","red"), main = title, xlab = "Read        Fak
}
```



NOTE: Dimension 1 corresponding first column in the dataset, i.e. length.
We see significant difference in Dimension 4&6 (Bottom margin & Diagonal Length), where the rest dimensions have difference in mean less than 1.

```
cor_df <- cor(swiss)
# Function to generate correlation plot
panel.corrgram <- function(x, y, z, subscripts, at, level = 0.9, label = FALSE, ...) {
     require("ellipse", quietly = TRUE)
     x <- as.numeric(x)[subscripts]
     y <- as.numeric(y)[subscripts]
     z <- as.numeric(z)[subscripts]
     zcol <- level.colors(z, at = at,  ...)
     for (i in seq(along = z)) {
         ell=ellipse(z[i], level = level, npoints = 50,
                     scale = c(.2, .2), centre = c(x[i], y[i]))
         panel.polygon(ell, col = zcol[i], border = zcol[i], ...)
     }
     if (label)
         panel.text(x = x, y = y, lab = 100 * round(z, 2), cex = 0.8,
                    col = ifelse(z < 0, "white", "black"))
 }
# generate correlation plot
print(levelplot(cor_df[seq(7,1), seq(7,1)], at = do.breaks(c(-1.01, 1.01), 20),
           xlab = NULL, ylab = NULL, colorkey = list(space = "top"), col.regions=rev(heat.colors(100)),
           scales = list(x = list(rot = 90)),
           panel = panel.corrgram, label = TRUE))
```

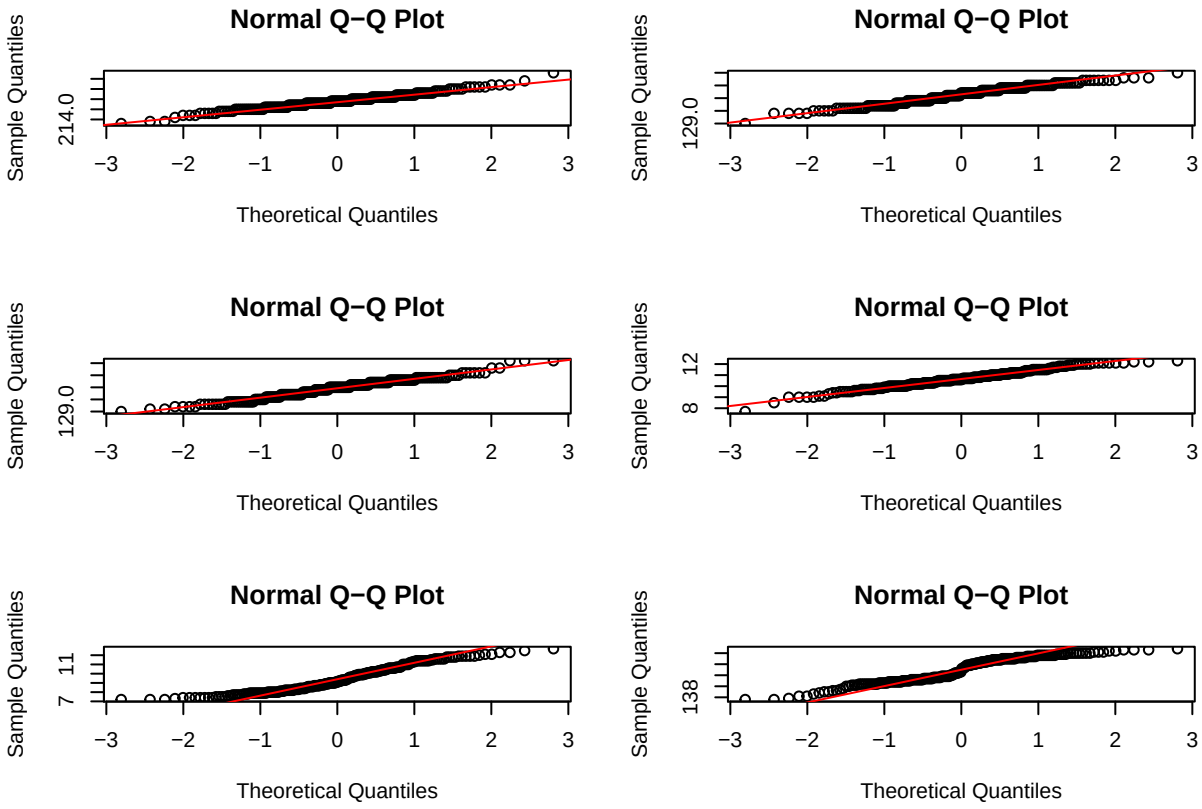


This level plot graph makes it very easy to investigate the correlations between variable. We can see the correlation between variables clearly. Some of these variables are even negatively correlated.
We can see that label is heavily correlated with "Diagonal" and "Bottom"

Using QQ-plots to confirm normal distribution:

```
par(mfrow = c(3,2))
qqnorm(swiss$Length)
qqline(swiss$Length, col='red')

qqnorm(swiss$Left)
qqline(swiss$Left, col = 'red')

qqnorm(swiss$Right)
qqline(swiss$Right, col = 'red')

qqnorm(swiss$Top)
qqline(swiss$Top, col = 'red')

qqnorm(swiss$Bottom)
qqline(swiss$Bottom, col = 'red')
```

```
qqnorm(swiss$Diagonal)
qqline(swiss$Diagonal, col = 'red')
```



We can see that the dataset is mostly normal except for some datapoints in Diagonal.

This agree with the conclusion from Lecture 10, where we performed two sample Hotelling's T-test, that the counterfeit notes can be distinguished from the genuine notes on at least one of the measurements.

Task 2: Divide into training and validation sets (which each must have some of the genuine and counterfeitnotes), implementing K-fold cross-validation.

```
# Change the Label
swiss$label[swiss$label == 0] = "real"
swiss$label[swiss$label == 1] = "fake"

# Turn label variable into type factor
swiss$label = as.factor(swiss$label)

# We choose K = 5 since it is a reasonable number that can evenly divide 200 and still
# have large enough observation.
K = 5
```

```r
# Specify the type of training method used
ctrlspecs = trainControl(method ="cv", number = K, savePredictions = "all", classProbs = TRUE)

# method = cross validation
# number = K = 5 number of folds
# savePredictions save all the predictions
# classProbs save class probability
```

Task 3: On each fold, classify using both LDA and logistic regression, i.e., fit both models on the training set and evaluate performance on the validation set.

LDA

Before we do LDA, we assume:
1. The data from group k has common mean vector $\underline{\mu}^{(k)}$, i.e.,

$$\mathbb{E}[x_{ij}^{(k)}] = \underline{\mu}_j^{(k)}.$$

(The m components of the vector correspond to the m variables.)
2. Homoskedasticity: The data from all groups have common covariance matrix , i.e.,

$$= \mathrm{Cov}[\underline{x}_i^{(k)}, \underline{x}_i^{(k)}]$$

for any record i, and the matrix does not depend on k (the group index).
3. Independence: The observations are independently sampled.
4. Normality: The data are multivariate normally distributed.

```r
# Setting seed to repeat analysis consistently
set.seed(123)

model_LDA1 = train(label~ ., data=swiss, method = "lda", trControl = ctrlspecs)
model_LDA1
```

```
## Linear Discriminant Analysis
##
## 200 samples
##   6 predictor
##   2 classes: 'fake', 'real'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 160, 160, 160, 160, 160
## Resampling results:
##
##   Accuracy  Kappa
##   0.995     0.99
```

The output of our training model of LDA returned accuracy of 99.5%. And kappa "rules of thumb" for interpretation if 0.99
Criterion for kappa value:
0.81-1.00 Perfect

0.61-0.80 Substantial
0.41-0.60 Moderate
0.21-0.40 Fair
0.00-0.20 Slight
<0.00 Poor

```
summary(model_LDA1)
```

```
##             Length Class      Mode
## prior        2     -none-     numeric
## counts       2     -none-     numeric
## means       12     -none-     numeric
## scaling      6     -none-     numeric
## lev          2     -none-     character
## svd          1     -none-     numeric
## N            1     -none-     numeric
## call         3     -none-     call
## xNames       6     -none-     character
## problemType  1     -none-     character
## tuneValue    1     data.frame list
## obsLevels    2     -none-     character
## param        0     -none-     list
```

From our QQ-plot, notice that "Diagonal" variable might not be normally distributed, which might violate one of our assumption.
And some assumptions need to be justified.

Logistic Regression

Assumptions: 1. The dependent variable to be binary and ordinal logistic regression requires the dependent variable to be ordinal
2. The observations to be independent of each other.
3. Logistic regression requires there to be little or no multicollinearity among the independent variables. This means that the independent variables should not be too highly correlated with each other. 4. Assume linearity of independent variables and log odds.
5. Requires a large sample size

```
# Setting seed to repeat analysis consistently
set.seed(123)

# Logistic Regression
model_glm1 = train(label ~ Length + Left + Right + Bottom + Diagonal, data=swiss, method="glm", family =
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

glm.fit returns error because some of these variables are NOT independent, e.g length and width will define diagonal. Since we discovered that only "Bottom" and "Diagonal" have higher correlated with label, and "Bottom" and "Diagonal" have rather high correlation between themselves. Only using "Bottom" is sufficient for Logistic Regression Model.

```
# Setting seed to repeat analysis consistently
set.seed(123)

# Logistic Regression
model_glm1 = train(label ~ Bottom, data=swiss, method="glm", family = binomial, trControl=ctrlspecs, ma
model_glm1
```

```
## Generalized Linear Model
##
## 200 samples
##   1 predictor
##   2 classes: 'fake', 'real'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 160, 160, 160, 160, 160
## Resampling results:
##
##   Accuracy  Kappa
##   0.91      0.82
```

The output of our training model of logistic regression returned accuracy of 91%. And kappa "rules of thumb" for interpretation if 0.82

```
summary(model_glm1)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.94462  -0.29027   0.05172   0.41782   2.34329
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  21.5922     2.9306   7.368 1.73e-13 ***
## Bottom       -2.3338     0.3197  -7.299 2.89e-13 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 277.26  on 199  degrees of freedom
## Residual deviance: 114.28  on 198  degrees of freedom
## AIC: 118.28
##
## Number of Fisher Scoring iterations: 6
```

Although we have really good prediction accuracy, we should take the result with some grain of salt due to 2 reasons:
1. The sample size is too small for performing logistic regression.
2. Some of the assumptions can not be verified with our current knowledge.

Task 4: Refine the six covariates using a factor model to reduce the dimension and remove any redundancy,re-running analysis on the factor scores.

We first fit PCA, then estimate the factor model with 2 factors via MLE.
Assumption for PCA:
1.variables are continuous
2.there's a linear relationship between all variables
3.sampling adequacy
4.no significant outliers

```
# PCA
pca_result = prcomp(data, scale =TRUE)
pca_result$rotation
```

```
##                   PC1         PC2         PC3        PC4        PC5         PC6
## Length    0.006987029 -0.81549497  0.01768066  0.5746173 -0.0587961  0.03105698
## Left     -0.467758161 -0.34196711 -0.10338286 -0.3949225  0.6394961 -0.29774768
## Right    -0.486678705 -0.25245860 -0.12347472 -0.4302783 -0.6140972  0.34915294
## Bottom   -0.406758327  0.26622878 -0.58353831  0.4036735 -0.2154756 -0.46235361
## Top      -0.367891118  0.09148667  0.78757147  0.1102267 -0.2198494 -0.41896754
## Diagonal  0.493458317 -0.27394074 -0.11387536 -0.3919305 -0.3401601 -0.63179849
```

```
eigen_val = pca_result$sdev^2
pve = eigen_val/sum(eigen_val)
pve
```

```
## [1] 0.49092637 0.21301396 0.14483876 0.07496145 0.04477948 0.03147998
```

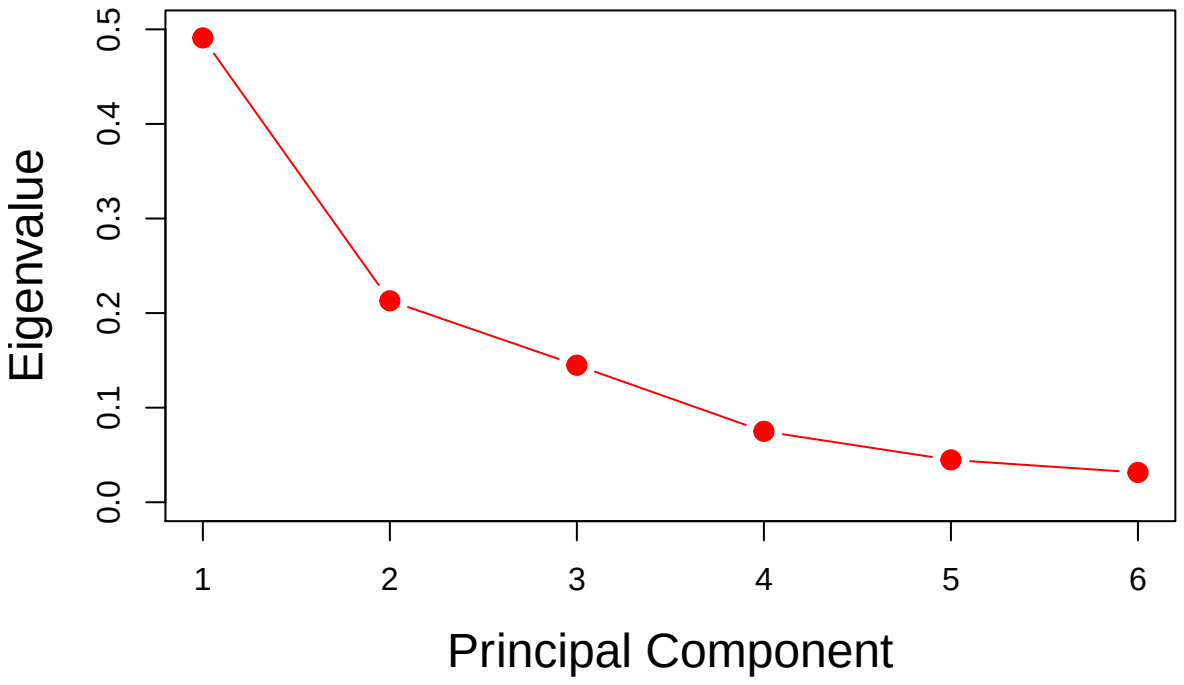
```
# MLE for factor model
n.factors = 2
fa_fit = factanal(data, n.factors, scores = "regression", rotation="varimax")
fa_fit$loadings
```

```
##
## Loadings:
##          Factor1 Factor2
## Length   -0.167   0.431
## Left      0.553   0.711
## Right     0.560   0.614
## Bottom    0.632   0.105
## Top       0.599
## Diagonal -0.995
##
##                Factor1 Factor2
## SS loadings      2.397   1.085
## Proportion Var   0.399   0.181
## Cumulative Var   0.399   0.580
```

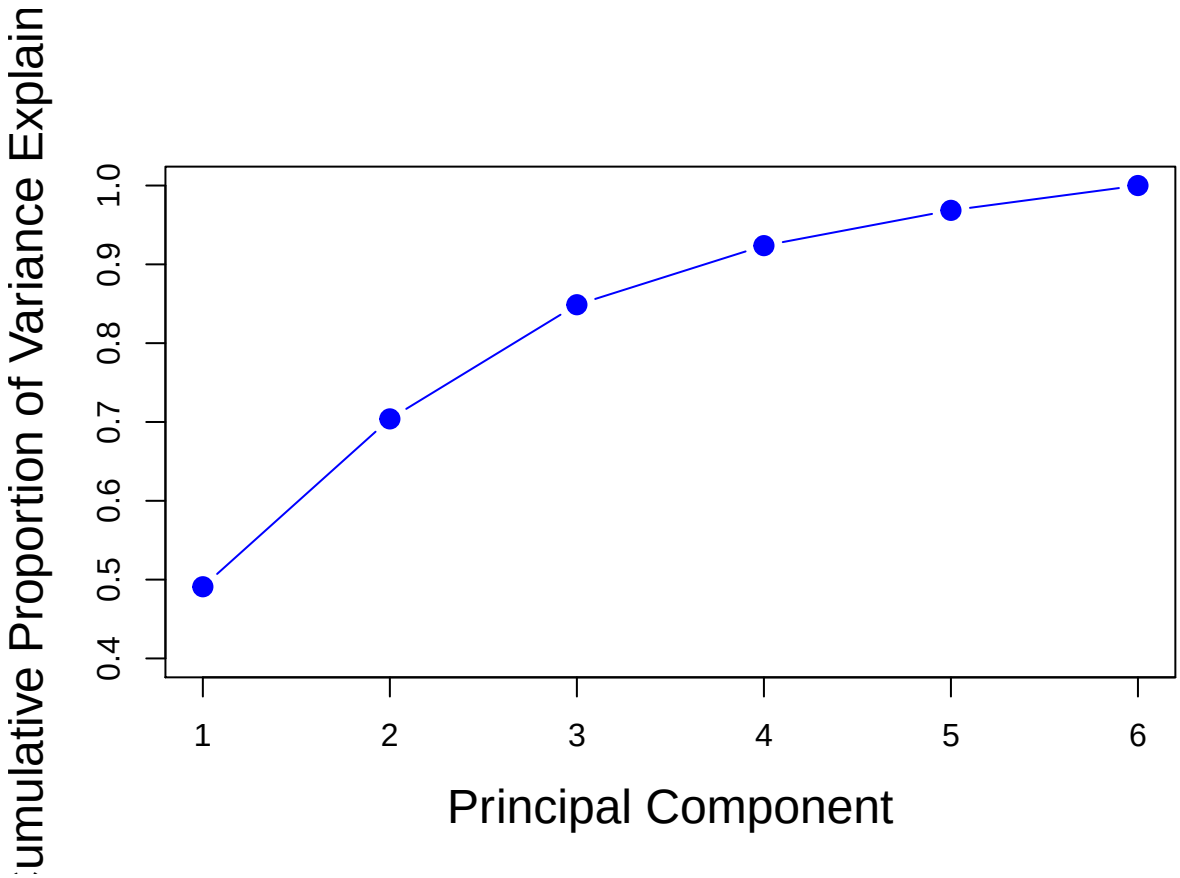
We assume for factor model:
1. factors and random errors have zero means
2. factors have identity covariance matrix
3. random errors have diagonal covariance matrix
4. factors and random errors are uncorrelated

Scree Plot To Decide On Dimension Reduction

```
plot(pve, xlab=" Principal Component ", ylab=" Eigenvalue ", ylim=c(0,0.5), xaxt="n",
type='b', col="red", cex=2, pch=20, cex.lab=1.5)
axis(1, at=c(1,2,3,4,5,6),labels=c(1,2,3,4,5,6))
```



```
plot(cumsum(pve), xlab=" Principal Component ",
ylab ="Cumulative Proportion of Variance Explained ",
ylim=c(0.4,1) , xaxt="n",type='b', col="blue", cex=2, pch=20, cex.lab=1.5)
axis(1, at=c(1,2,3,4,5,6,7),labels=c(1,2,3,4,5,6,7))
```



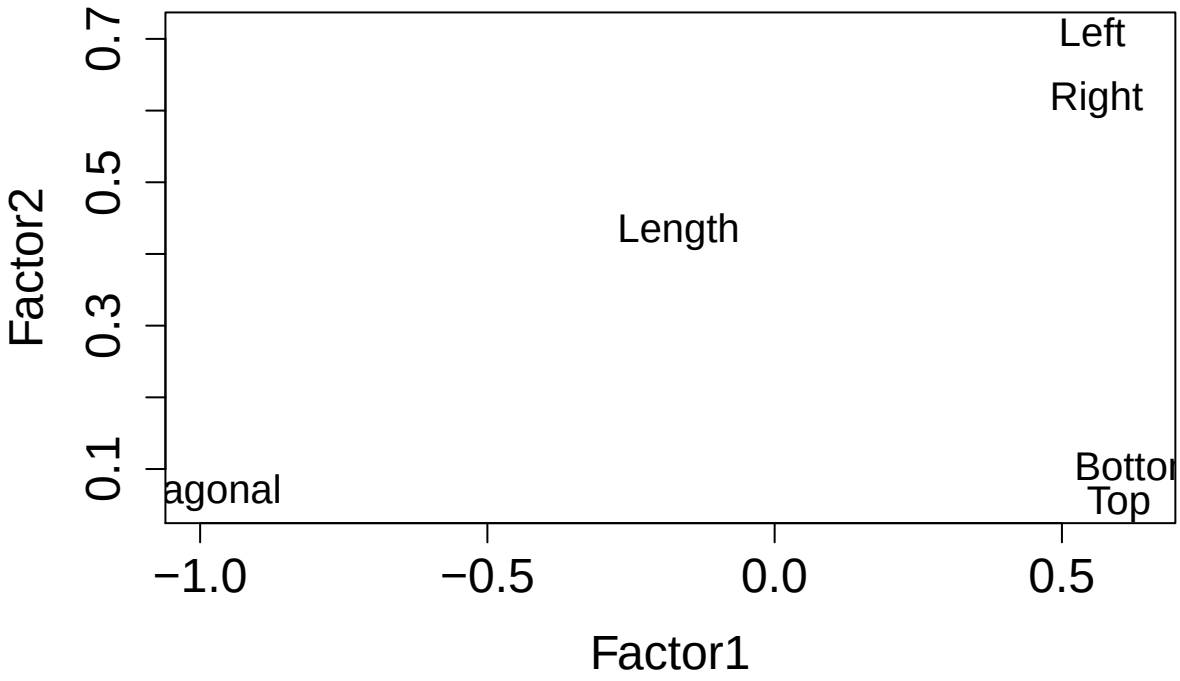
We choose number of components to be 2. Not only 2 components are enough to cover more than 70% variance, which is higher than 50%, but also far less dimensions than 6.

Examine The Factor Loading

```
loading <- fa_fit$loadings[,1:2]
loading
```

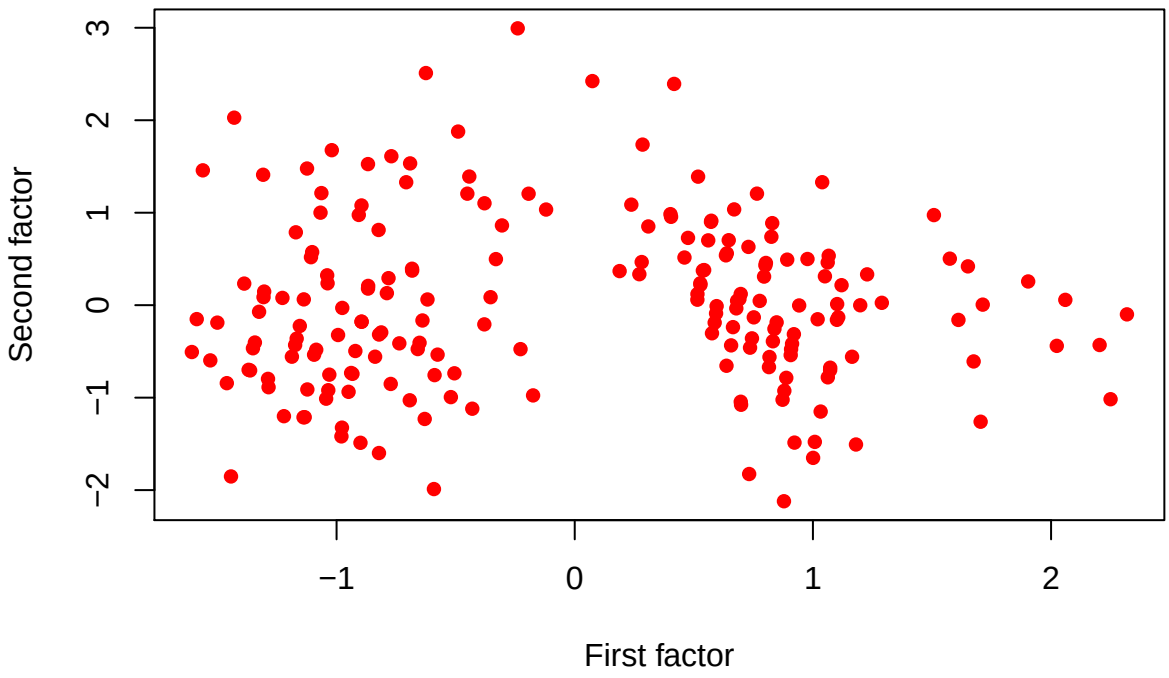
```
##              Factor1    Factor2
## Length    -0.1670846 0.43050687
## Left       0.5527990 0.71052785
## Right      0.5603285 0.61426694
## Bottom     0.6323679 0.10476254
## Top        0.5991447 0.05087066
## Diagonal  -0.9952948 0.06627958
```

```
plot(loading,type="n",cex.axis=1.5,cex.lab=1.5) # set up plot
text(loading,labels=names(data),cex=1.25)
```



Factor 1 assigns high loading to Bottom, Top, left,right, but a negative loading to Diagonal.
Factor 2 assigns high loading to Length, Left, and Right.

```
score = fa_fit$scores
plot(score[, 1], score[, 2], pch = 16,  xlab="First factor", ylab="Second factor", col="red")
```



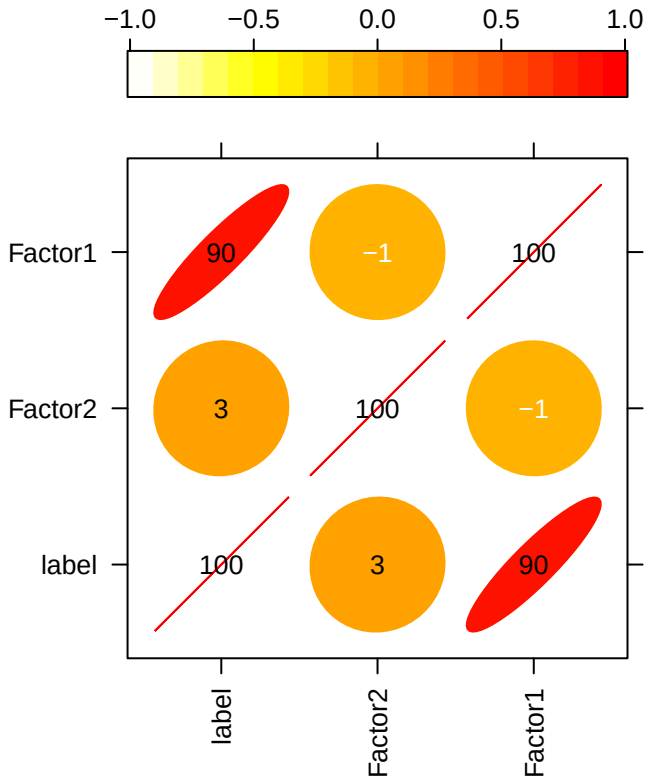
Note that real note contain factor 1 > 0; fake note contain factor 1 <0.

```
swiss_clean = cbind(fa_fit$scores, label)
head(swiss_clean)
```

```
##        Factor1    Factor2 label
## BN1 -0.2394320  2.9936843     0
## BN2 -1.0944004 -0.5357972     0
## BN3 -1.5006479 -0.1888711     0
## BN4 -1.3429929 -0.4038795     0
## BN5 -1.1875631 -0.5570299     0
## BN6 -0.6248881  2.5100863     0
```

```
cor_df <- cor(swiss_clean)
# Function to generate correlation plot
panel.corrgram <- function(x, y, z, subscripts, at, level = 0.9, label = FALSE, ...) {
     require("ellipse", quietly = TRUE)
     x <- as.numeric(x)[subscripts]
     y <- as.numeric(y)[subscripts]
     z <- as.numeric(z)[subscripts]
     zcol <- level.colors(z, at = at,  ...)
     for (i in seq(along = z)) {
         ell=ellipse(z[i], level = level, npoints = 50,
                     scale = c(.2, .2), centre = c(x[i], y[i]))
         panel.polygon(ell, col = zcol[i], border = zcol[i], ...)
```

```
    }
    if (label)
        panel.text(x = x, y = y, lab = 100 * round(z, 2), cex = 0.8,
                   col = ifelse(z < 0, "white", "black"))
 }
# generate correlation plot
print(levelplot(cor_df[seq(3,1), seq(3,1)], at = do.breaks(c(-1.01, 1.01), 20),
           xlab = NULL, ylab = NULL, colorkey = list(space = "top"), col.regions=rev(heat.colors(100)),
           scales = list(x = list(rot = 90)),
           panel = panel.corrgram, label = TRUE))
```




```
# Change the Label
swiss_clean = as.data.frame(swiss_clean)
swiss_clean$label[swiss_clean$label == 0] = "real"
swiss_clean$label[swiss_clean$label == 1] = "fake"

# Turn label variable into type factor
swiss_clean$label = as.factor(swiss_clean$label)

# We choose K = 5 since it is a reasonable number that can evenly divide 200 and still
# have large enough observation.
K = 5

# Specify the type of training method used
ctrlspecs = trainControl(method ="cv", number = K, savePredictions = "all", classProbs = TRUE)
```

```
# method = cross validation
# number = K = 5 number of folds
# savePredictions save all the predictions
# classProbs save class probability
```

Now we train the LDA model with reduced dimension dataset under the same assumptions.

```
# Setting seed to repeat analysis consistently
set.seed(123)

model_LDA2 = train(label~ ., data=swiss_clean, method = "lda", trControl = ctrlspecs)
model_LDA2
```

```
## Linear Discriminant Analysis
##
## 200 samples
##   2 predictor
##   2 classes: 'fake', 'real'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 160, 160, 160, 160, 160
## Resampling results:
##
##   Accuracy  Kappa
##   0.99      0.98
```

The output of our training model of LDA returned accuracy of 99%. And kappa "rules of thumb" for interpretation if 0.98.
We see no significant improvement compare to our original model.

Now we train the logistic regression model with reduced dimension dataset under the same assumptions.

```
# Setting seed to repeat analysis consistently
set.seed(123)

# Logistic Regression
model_glm2 = train(label ~ Factor1 + Factor2, data=swiss_clean, method="glm", family = binomial, trCont
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
model_glm2
```

```
## Generalized Linear Model
##
## 200 samples
##   2 predictor
##   2 classes: 'fake', 'real'
```

```
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 160, 160, 160, 160, 160
## Resampling results:
##
##   Accuracy  Kappa
##   0.99      0.98
```

It seems like Factor1 & Factor2 are redundant, we can use only one of them to train our model

```
# Setting seed to repeat analysis consistently
set.seed(123)

# Logistic Regression
model_glm2 = train(label ~ Factor1, data=swiss_clean, method="glm", family = binomial, trControl=ctrlsp
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
model_glm2
```

```
## Generalized Linear Model
##
## 200 samples
##   1 predictor
##   2 classes: 'fake', 'real'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 160, 160, 160, 160, 160
## Resampling results:
##
##   Accuracy  Kappa
##   0.99      0.98
```

```
fa_fit$scores[97:104,]
```

```
##          Factor1    Factor2
## BN97  -0.4509183  1.2057581
## BN98  -0.6927138 -1.0287343
## BN99  -0.8673469  0.2084003
## BN100 -0.6591588 -0.4745896
## BN101  0.5875324 -0.1889459
## BN102  0.8920605  0.4922324
## BN103  0.2712952  0.3347769
## BN104  0.2375682  1.0868553
```

Follow a close inspection on factor score, we see that real bank note corresponding to a negative Factor1 value, fake bank note corresponding to a positive Factor1 value. It is understandable that glm.fit gave a warning message since Factor1 value alone can determine the authenticity of the bank note. Thus probabilities of logistic regression model is numerically 1.

Our accuracy = 99% and kappa = 0.098. Values increased slightly compare to the dataset before factor analysis.

```
summary(model_glm2)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1030  -0.0525   0.0005   0.0211   3.4572
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.4559     0.7060   0.646    0.518
## Factor1      -8.5521     2.0172  -4.239 2.24e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 277.259  on 199  degrees of freedom
## Residual deviance:  17.929  on 198  degrees of freedom
## AIC: 21.929
##
## Number of Fisher Scoring iterations: 9
```

We see a slight increase in accuracy in Logistic Regression model and relatively the same value for LDA after factor analysis.

Justification for assumptions

LDA
It is hard to justify all the assumptions but normality is violated for variable "Diagonal" according to our QQ-plot.

Logistic Regression
We have a rather small sample size of 200, and dimension variables are not linearly independent as diagonal can be calculated by length and left or right via Pythagorean Theorem.
Other assumptions are hard to justify.

PCA
PCA assume linear relationship between all variables, which is clearly not the case for our dataset. Diagonal does NOT have linear relationship between length, left and right.

Factor Model

```
colMeans(fa_fit$scores)
```

```
##       Factor1       Factor2
## -6.408832e-15  6.341316e-15
```

```
cov(fa_fit$scores)
```

```
##              Factor1      Factor2
## Factor1  0.994641768 -0.008842023
## Factor2 -0.008842023  0.819359620
```

Covariance matrix is not quite equal to identity matrix.

Conclusion

In this project we performed training of 2 models, LDA and Logistic Regression, on Swiss Bank Notes dataset. Then we attempt to do Factor Analysis in order to reduction dimensions of dataset to see if there is any improvement in prediction accuracy.
It is not waste of time doing factor analysis, we saw increase in accuracy about 9% for Logistic Regression and no significant improvement for LDA. But the increase in accuracy is relatively small with the fact that we have fairly high accuracy of 91% before factor analysis. We can save time by using initial model and still having high accuracy. But going from 91% to almost 100% accuracy is still a great improvement. Depends on the circumstances, factor analysis can be applied to increase accuracy.
We suggest using LDA for initial dataset since it has the least violation, easy to setup and have great accuracy of 99.5%. Since there are violations for models, PCA, and factor model. We should take the model accuracy with some grain of salt.
Keep in mind that we are working with a very small sample size. All the results may change if we have a sufficiently large sample size. With a larger sample size, we can divide dataset into training group and testing group, and test our model after training to future comfirm the true accuracy of our model. We highly recommend to repeat this analysis with a larger sample.