Math 189: Final Project

Swiss Bank Notes

Working alone, prepare a R Markdown Notebook report based on examining the Swiss bank notes dataset (available from GitHub). The dataset contains six variables measured on 100 genuine and 100 counterfeit old Swiss 1000-franc bank notes:

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

Introduction

In this project, we attempt to determine whether or not a note is counterfeit using supervised learning. There is quite a bit in this project, but it can be reduced down to first visualizing the data and noticing correlation. Other things we may look for are normality and outliers. We then randomly split the data into k folds (here we used 5). Then, we conduct Linear Discriminant Analysis and Logistic Regression using each of the folds. After that, we then reduce dimensions using factor analysis. We then re-conduct LDA and Logistic Regression on the factors to see if that helped any. In essence, we are using the geometric dimensions on a Swiss bank note in an attempt to determine if its Counterfeit. Then, we try to reduce the number of variables to certain factors that account for the majority of variability. We conduct multiple times in different ways all in an attempt to find the best model to conduct this research. Of course, other things will be a part of our decision such as interpretability, ease of computation, simplicity, and assumptions needed.

Data

This dataset comprises of 6 variable lengths of 200 Swiss bank notes: Length, Left, Right, Bottom, Top, and Diagonal. The first 100 are genuine and the second 100 are counterfeit which we added into the dataset with the binary variable "Counterfeit". All are measured in millimeters. The data comes from Flury, B. and Riedwyl, H. (1988). Multivariate Statistics: A practical approach. London: Chapman & Hall, Tables 1.1 and 1.2, pp. 5-8. https://rdrr.io/cran/mclust/man/banknote.html

Methods and Analysis:

As per usual, there would be way too much information put into this single section, and thus, we will write the methods and analysis above each r-code.

Here, we load in the dataset and add in a column "Counterfeit" with 0 for the first 100 (genuine notes) and 1 for the second 100 (counterfeit notes).

```r
"C:\\Users\\Mark's PC\\Desktop\\Math 189\\ma189-main\\Data\\SBN.txt" = paste0(getwd(), "/SBN.txt")
notes = read.table("C:\\Users\\Mark's PC\\Desktop\\Math 189\\ma189-main\\Data\\SBN.txt", header = TRUE)
notes$Counterfeit = 0
notes[1:100,7] = 0
notes[101:200,7] = 1
head(notes)
```

```
##     Length  Left Right Bottom  Top Diagonal Counterfeit
## 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
```

Here we simply calculate the mean, variance, covariance, and correlation of the variables to begin visualizing the data.

```r
Mu = colMeans(notes[,1:6])
Mu
```

```
##   Length     Left    Right   Bottom      Top Diagonal
## 214.8960 130.1215 129.9565   9.4175  10.6505 140.4835
```

```r
Var = apply(notes[,1:6], 2, var)
Var
```

```
##    Length      Left     Right    Bottom       Top  Diagonal
## 0.1417930 0.1303394 0.1632741 2.0868781 0.6447234 1.3277163
```

```r
Cov = cov(notes[,1:6])
Cov
```

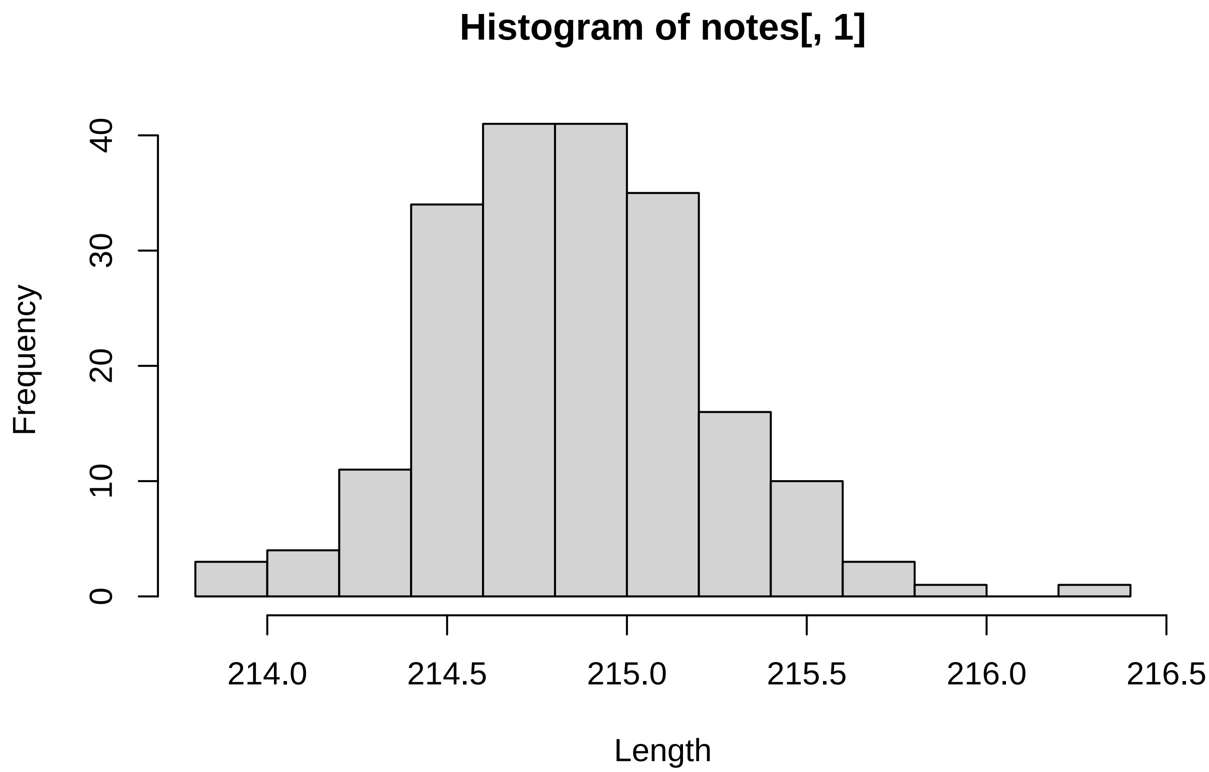
```
##               Length        Left       Right     Bottom        Top    Diagonal
## Length    0.14179296  0.03144322  0.02309146 -0.1032462 -0.0185407  0.08430553
## Left      0.03144322  0.13033945  0.10842739  0.2158028  0.1050394 -0.20934196
## Right     0.02309146  0.10842739  0.16327412  0.2841319  0.1299967 -0.24047010
## Bottom   -0.10324623  0.21580276  0.28413191  2.0868781  0.1645389 -1.03699623
## Top      -0.01854070  0.10503945  0.12999673  0.1645389  0.6447234 -0.54961482
## Diagonal  0.08430553 -0.20934196 -0.24047010 -1.0369962 -0.5496148  1.32771633
```

```r
Cor = cor(notes[,1:6])
Cor
```

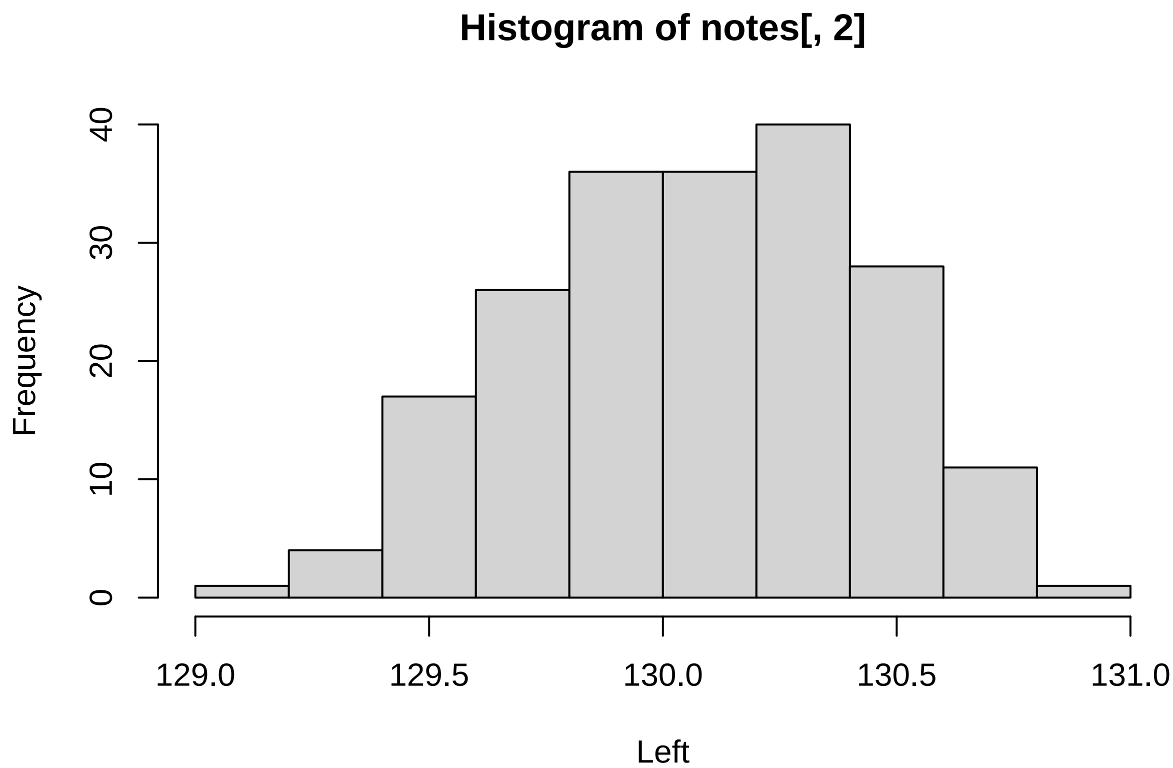
```
##               Length       Left      Right     Bottom         Top   Diagonal
## Length    1.00000000  0.2312926  0.1517628 -0.1898009 -0.06132141  0.1943015
## Left      0.23129257  1.0000000  0.7432628  0.4137810  0.36234960 -0.5032290
## Right     0.15176280  0.7432628  1.0000000  0.4867577  0.40067021 -0.5164755
## Bottom   -0.18980092  0.4137810  0.4867577  1.0000000  0.14185134 -0.6229827
## Top      -0.06132141  0.3623496  0.4006702  0.1418513  1.00000000 -0.5940446
## Diagonal  0.19430146 -0.5032290 -0.5164755 -0.6229827 -0.59404464  1.0000000
```

We create histograms with each of the variables. This will not show correlation, but it will show normality. Notice, that not all the graphs are completely normal. The histogram of the variable "Bottom" seems skewed, and the histogram of the variable "Diagonal" is bimodal. This could be from the fact that there are half genuine and half counterfeit, or could be from other factors.

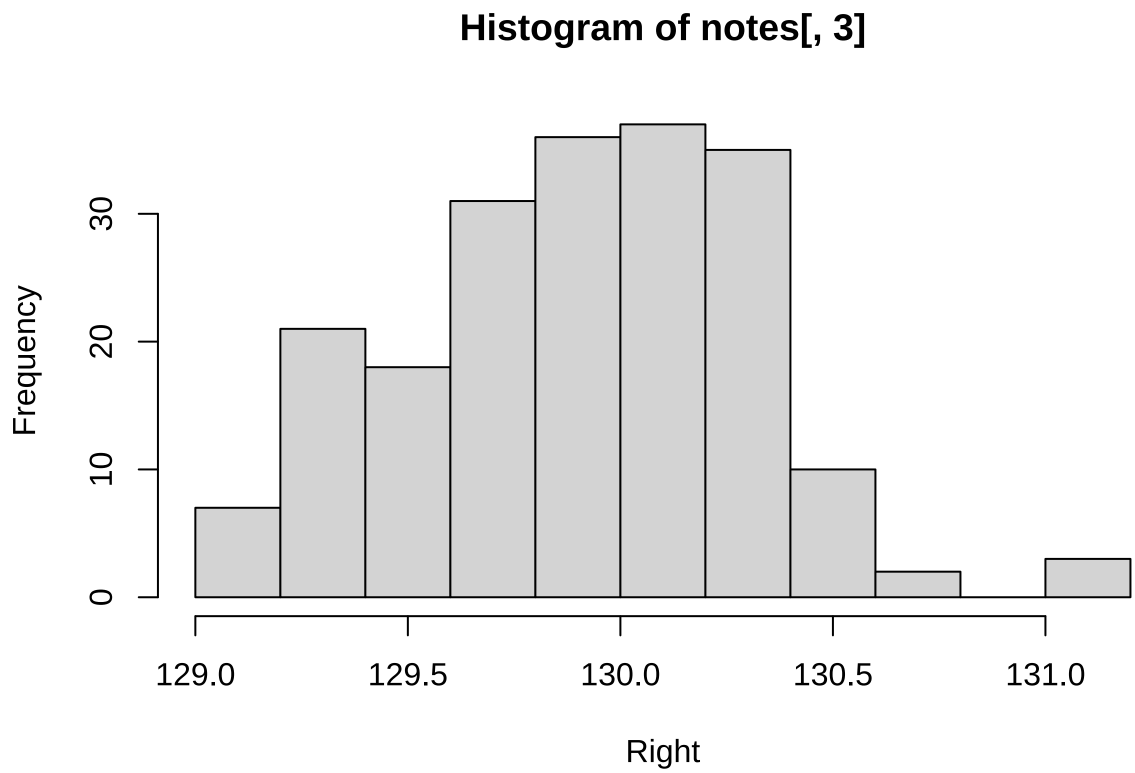
```
hist(notes[,1],xlab = "Length")
```



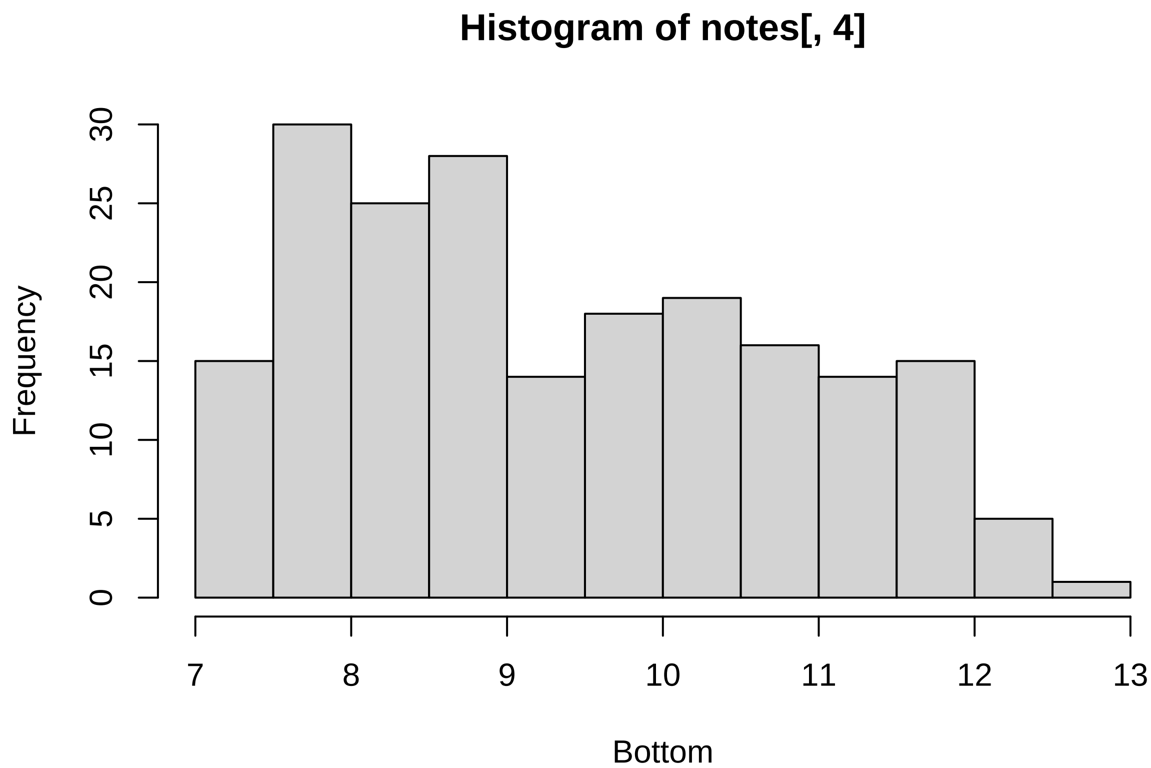

```
hist(notes[,2],xlab = "Left")
```



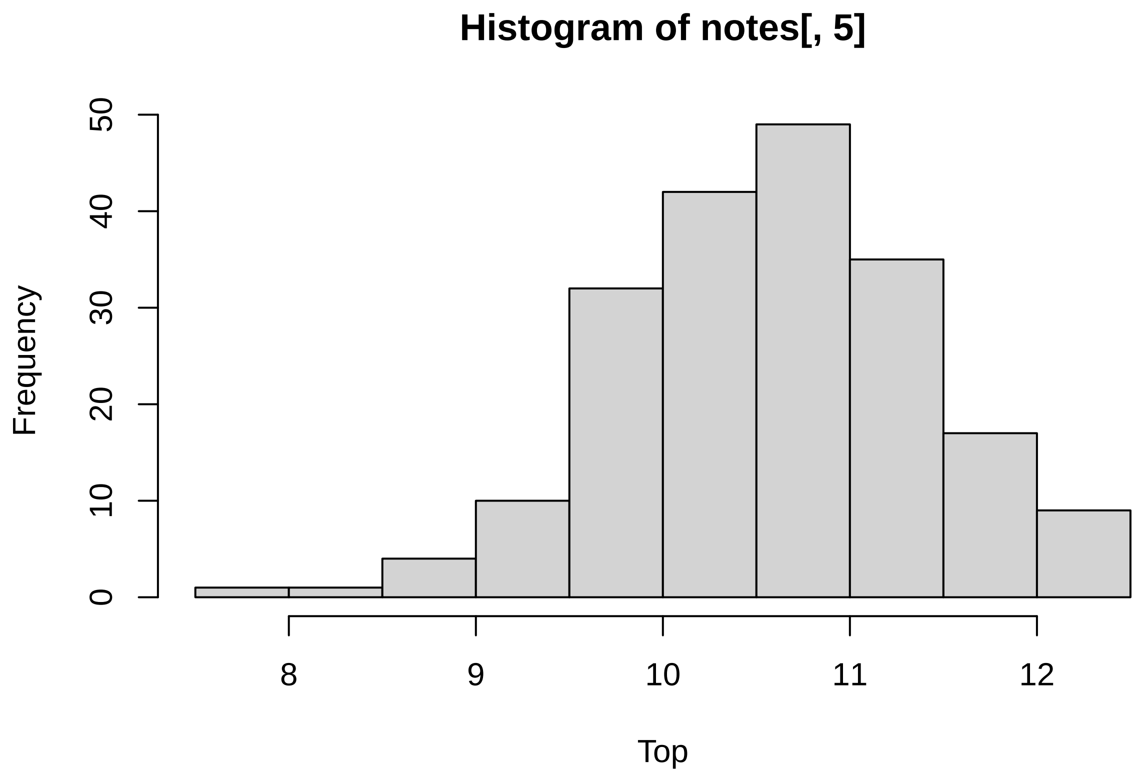

```
hist(notes[,3],xlab = "Right")
```



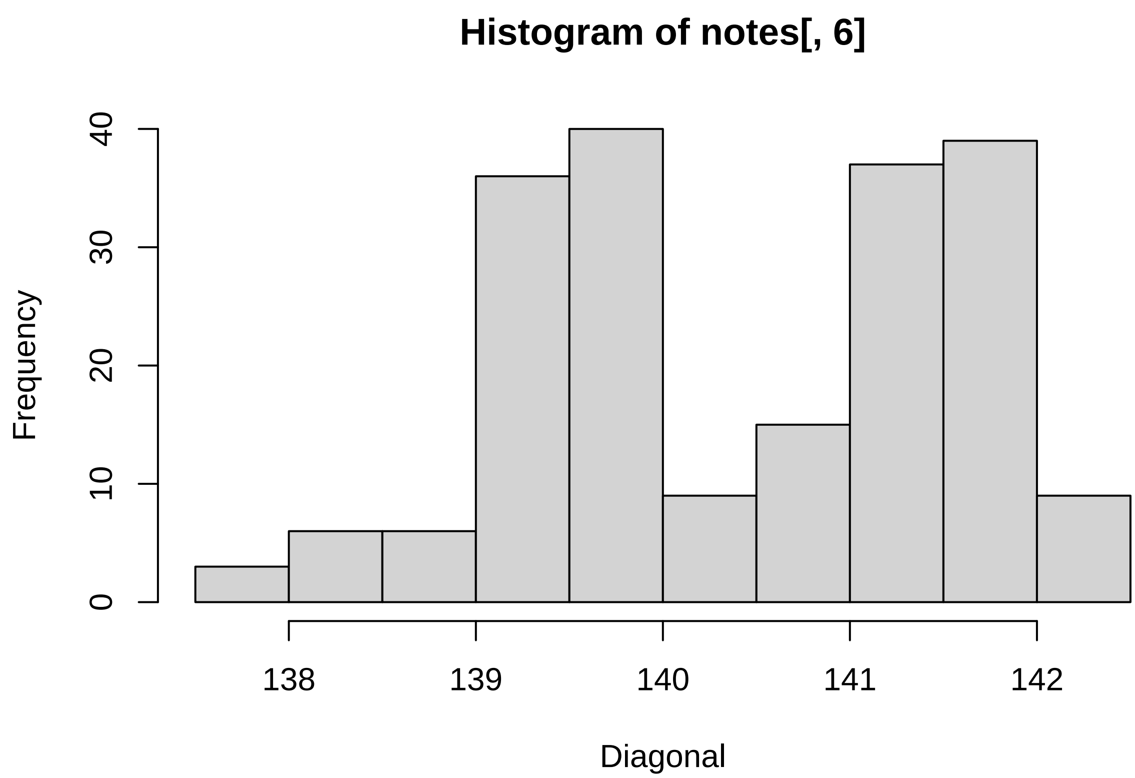

```
hist(notes[,4],xlab = "Bottom")
```




```r
hist(notes[,5],xlab = "Top")
```




```r
hist(notes[,6],xlab = "Diagonal")
```



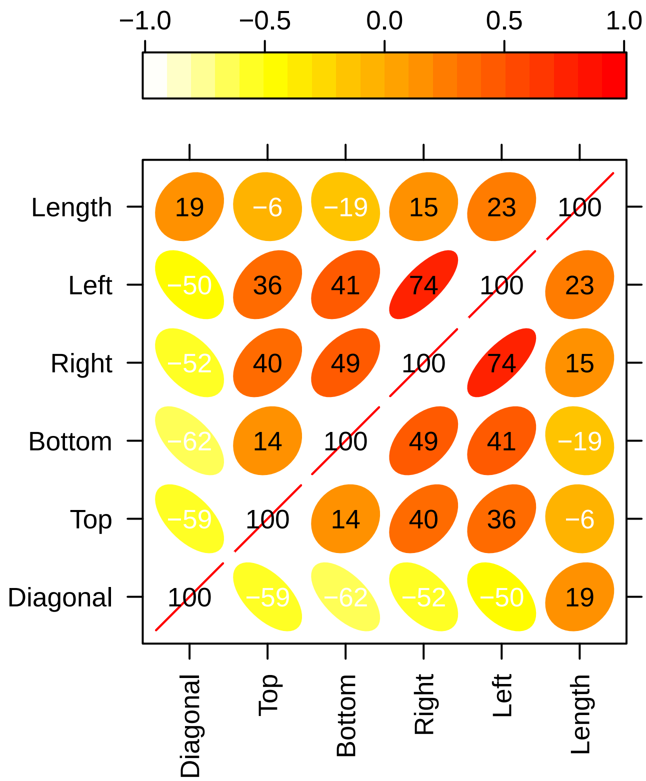
Here, we create a correlation plot to quite easily view the correlation between variables. The graph ranges from white(negative correlation) to orange(zero correlation) to red(positive correlation). We notice that diagonal is quite negatively correlated with all variables besides length. We also notice that Left width and Right width are extremely positively correlated.

```
library(lattice)
library(ellipse)
```

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

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

```
cor_nut <- cor(notes)

# 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)) {
```

```r
        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_nut[seq(6,1), seq(6,1)], 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))
```



Here we put the bank notes in random order, so that each fold will have a combination of counterfeit and genuine bank notes. We also set a seed so that our results will give the same output everytime we run the code.

```r
set.seed(42)
index = sample(200,200)
index
```

```
##   [1]  49  65 153  74 146 122 200 128  47  24  71 100  89 165 110  20 154 114
##  [19] 111 131  41 188  27 164 109   5 162  92 104   3  58  42 191 158  43 143
```

```
##  [37] 150 170 136  36  68 196 176 173   4  99 184 183   6 134 130 116 171 118
##  [55]   2 102 138  40 175  33 103 167  73  76   9  35  16 101  69 147 177  82
##  [73] 168 113  18 132 186 172  55 187  21 189  57 119 140 169 126  91  13  53
##  [91]  54  83  32  80  60  29  81 144  85 166 163  72 105 195  38   1 112  78
## [109] 142 149  97 151 133 115  87 181  98  25  63 108  14 152 192  88  62  37
## [127]  31  34  79  96 155  15 127  86 106  12  64  26 180  95 159  56 193 160
## [145] 139  28  77 107 145  66 197 123 178  10  90  75  93 157 135 182 137  46
## [163] 141  45  67 129  50 194  17  48  52 185 156  84  11 190 174  59  94  23
## [181]  30  61 120  19 199  51   8 148 124 198  22 179   7  70 125 161 117  39
## [199]  44 121
```

```
Empty = notes
for(i in 1:200){
  Empty[i,] = notes[index[i],]
}
notes = Empty
```

Here, we simply create the 5 different folds each comprising of 40 of the notes. When we run LDA and LogReg later, 1 of these folds will be the validation and the remaining 4 will be the training set. We will run it 5 times for each of the folds.

```
fold1 = notes[1:40,]
fold2 = notes[41:80,]
fold3 = notes[81:120,]
fold4 = notes[121:160,]
fold5 = notes[161:200,]
```

Assumptions for LDA:

1. The data from group k has common mean vector. We may simply assume this. One way to test is to take random samples from the data and check if the means are equivalent.

2. Homoskedasticity: The data from all groups have common variance. We will demonstrate one variance test to show this, and for the rest it will be assumed.

```
var.test(notes$Length,notes$Left,alternative = "two.sided")
```

```
##
##  F test to compare two variances
##
## data:  notes$Length and notes$Left
## F = 1.0879, num df = 199, denom df = 199, p-value = 0.553
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.8232889 1.4374918
## sample estimates:
## ratio of variances
##           1.087875
```

3. The data from each group is independently sampled. This is quite difficult to prove, so we will just assume this.

4. The data are normally distributed. This is somewhat of a problem, because if you look at the histograms, the values for “Diagonal” are bimodal, and the values for “Bottom” are skewed. Because of this, we might prefer using a different method of analysis such as logistic regression.

This is the full function for running Linear Discriminant Analysis. I won’t go into the whole process, as we have done it before, and the names are fairly self-explanatory. We put it into a function for ease of use; otherwise, we would have to type this code 5 separate times.

```
LDA = function(train,val){
  Counter1_TR = train[train$Counterfeit == "1",1:6]
  Counter0_TR = train[train$Counterfeit == "0",1:6]
  n1_TR = dim(Counter1_TR)[1]
  n0_TR = dim(Counter0_TR)[1]
  N_TR = n1_TR + n0_TR
  Counter1_p = n1_TR/N_TR
  Counter0_p = n0_TR/N_TR
  Counter1_mu = colMeans(Counter1_TR)
  Counter0_mu = colMeans(Counter0_TR)
  rbind(Counter1_mu,Counter0_mu)
  Counter1_S = cov(Counter1_TR)
  Counter0_S = cov(Counter0_TR)
  S_pool = ((n1_TR-1)*Counter1_S + (n0_TR-1)*Counter0_S)/(n1_TR + n0_TR - 1)
  S_inv = solve(S_pool)
  Counter1_alpha = -0.5*t(Counter1_mu) %*% S_inv %*% Counter1_mu +    log(Counter1_p)
  Counter0_alpha = -0.5*t(Counter0_mu) %*% S_inv %*% Counter0_mu + log(Counter0_p)
  Counter01_alpha = c(Counter1_alpha,Counter0_alpha)
  Counter01_alpha
  Counter1_beta = S_inv %*% Counter1_mu
  Counter0_beta = S_inv %*% Counter0_mu
  Counter01_beta = cbind(Counter1_beta,Counter0_beta)
  Counter01_beta

  prediction = c()
Counter1_dvec = c()
Counter0_dvec = c()
label = c("1","0")
for(i in 1:nrow(val)){
x = t(val[i,1:6])
Counter1_d = Counter1_alpha + t(Counter1_beta) %*% x
Counter0_d = Counter0_alpha + t(Counter0_beta) %*% x
dvec = c(Counter1_d,Counter0_d)
prediction = append(prediction, label[which.max(dvec)])
Counter1_dvec = append(Counter1_dvec,Counter1_d)
Counter0_dvec = append(Counter0_dvec,Counter0_d)
}
val$prediction = prediction
val
}
```

This is simply a function that, after we compute the prediction column for the validation set, we can calculate the number we go correct and incorrect.

```r
NumCorrect = function(test){
Correct1 = 0
Incorrect1 = 0
Correct0 = 0
Incorrect0 = 0
for(i in 1:40){
  if(test[i,7] == 1){
    if(test[i,7] == test[i,8]){
      Correct1 = Correct1 + 1
    }
    if(test[i,7] != test[i,8]){
      Incorrect1 = Incorrect1 + 1
    }
  }
  if(test[i,7] == 0){
    if(test[i,7] == test[i,8]){
      Correct0 = Correct0 + 1
    }
    if(test[i,7] != test[i,8]){
      Incorrect0 = Incorrect0 + 1
    }
  }
}
Table1 = rbind(Correct1 + Incorrect1,Correct1,Incorrect1)
Table0 = rbind(Correct0 + Incorrect0,Correct0,Incorrect0)
Table_full = cbind(Table1,Table0)
colnames(Table_full) = c("Counterfeit1","Counterfeit0")
rownames(Table_full) = c("Number of Observations","Correct","Incorrect")
Table_full
}
```

The next 5 r-blocks are computing LDA on each of the folds with the other 4 being the training set. We then compute the number of correct predictions for each one, as we will use this to create a concise table of the model prediction accuracy at the end. We will demonstrate the first prediction table only to avoid redundancy. If you like, you can verify the prediction results.

```r
val1 = fold1
train1 = rbind(fold2,fold3,fold4,fold5)
test1 = LDA(train1,val1)
test1
```

```
##      Length  Left Right Bottom  Top Diagonal Counterfeit prediction
## BN1   214.6 129.7 129.8    7.9 10.3    141.1           0          0
## BN2   215.0 130.0 129.8    8.6 10.6    141.5           0          0
## BN3   214.6 129.7 129.3   10.4 11.0    139.3           1          1
## BN4   214.4 129.9 129.6    7.5 10.5    141.8           0          0
## BN5   214.9 130.6 130.4   11.9 10.5    139.8           1          1
## BN6   215.1 130.6 130.3   12.3 10.2    139.6           1          1
## BN7   214.3 129.9 129.9   10.2 11.5    139.6           1          1
## BN8   215.5 130.7 130.3   10.2 11.8    140.0           1          1
## BN9   214.8 129.9 129.7    8.3 10.2    141.5           0          0
## BN10  215.7 130.2 130.0    8.7 10.0    141.6           0          0
## BN11  213.8 129.8 129.5    8.4 11.1    140.9           0          0
```

```
## BN12  214.7 130.0 129.4    7.8 10.0    141.2           0           0
## BN13  214.9 130.3 129.9    7.4 11.2    141.5           0           0
## BN14  214.3 130.3 130.0   11.4 10.5    139.8           1           1
## BN15  215.2 130.6 130.8   10.4 11.2    140.3           1           1
## BN16  214.7 130.2 129.9    8.6 10.0    141.9           0           0
## BN17  214.5 130.1 130.1   12.1 10.3    139.4           1           1
## BN18  214.9 130.4 129.9   11.4 11.0    139.9           1           1
## BN19  215.2 130.4 130.3    8.0 11.5    139.2           1           1
## BN20  214.3 130.2 130.0   10.7 10.5    139.8           1           1
## BN21  214.4 129.8 129.2    8.9  9.4    142.3           0           0
## BN22  214.8 130.0 129.7   11.4 10.6    139.2           1           1
## BN23  215.5 130.2 130.1    8.9  9.8    142.4           0           0
## BN24  214.7 130.1 130.2   11.6 10.9    139.1           1           1
## BN25  215.0 130.2 129.9   10.0 11.9    139.4           1           1
## BN26  215.0 129.6 129.7   10.4  7.7    141.8           0           0
## BN27  214.9 130.5 130.1    9.9 10.2    138.1           1           1
## BN28  215.4 130.0 129.9    8.5  9.7    141.4           0           0
## BN29  215.0 130.4 130.6    9.9 10.9    140.3           1           1
## BN30  214.8 129.7 129.7    8.7  9.6    142.2           0           0
## BN31  215.0 129.6 129.4    8.8  9.0    141.1           0           0
## BN32  214.8 130.1 129.6    8.8  9.9    140.9           0           0
## BN33  215.1 130.2 129.8   10.2 12.0    139.4           1           1
## BN34  214.4 130.1 130.0   11.3 10.7    139.2           1           1
## BN35  214.9 129.6 129.4    9.3  9.0    141.7           0           0
## BN36  214.6 130.2 130.4   11.2 10.7    139.9           1           1
## BN37  214.9 129.9 130.0    9.9 12.3    139.4           1           1
## BN38  214.9 130.0 129.9   11.4 10.9    139.7           1           1
## BN39  214.8 130.1 130.1   11.9 11.1    139.5           1           1
## BN40  214.6 130.2 130.2    9.4  9.7    141.8           0           0
```

```r
NumCorrect(test1)
```

```
##                          Counterfeit1 Counterfeit0
## Number of Observations             22           18
## Correct                            22           18
## Incorrect                           0            0
```

```r
LDA_fold1_correct = (NumCorrect(test1)[2,1]+NumCorrect(test1)[2,2])/(NumCorrect(test1)[1,1]+NumCorrect(t
```

```r
val2 = fold2
train2 = rbind(fold1,fold3,fold4,fold5)
test2 = LDA(train2,val2)
NumCorrect(test2)
```

```
##                          Counterfeit1 Counterfeit0
## Number of Observations             24           16
## Correct                            24           16
## Incorrect                           0            0
```

```r
LDA_fold2_correct = (NumCorrect(test2)[2,1]+NumCorrect(test2)[2,2])/(NumCorrect(test2)[1,1]+NumCorrect(t
```

```r
val3 = fold3
train3 = rbind(fold1,fold2,fold4,fold5)
test3 = LDA(train3,val3)
NumCorrect(test3)
```

```
##                          Counterfeit1 Counterfeit0
## Number of Observations             18           22
## Correct                            18           22
## Incorrect                           0            0
```

```r
LDA_fold3_correct = (NumCorrect(test3)[2,1]+NumCorrect(test3)[2,2])/(NumCorrect(test3)[1,1]+NumCorrect(t
```

```r
val4 = fold4
train4 = rbind(fold1,fold2,fold3,fold5)
test4 = LDA(train4,val4)
NumCorrect(test4)
```

```
##                          Counterfeit1 Counterfeit0
## Number of Observations             18           22
## Correct                            18           22
## Incorrect                           0            0
```

```r
LDA_fold4_correct = (NumCorrect(test4)[2,1]+NumCorrect(test4)[2,2])/(NumCorrect(test4)[1,1]+NumCorrect(t
```

```r
val5 = fold5
train5 = rbind(fold1,fold2,fold3,fold4)
test5 = LDA(train5,val5)
NumCorrect(test5)
```

```
##                          Counterfeit1 Counterfeit0
## Number of Observations             18           22
## Correct                            18           21
## Incorrect                           0            1
```

```r
LDA_fold5_correct = (NumCorrect(test5)[2,1]+NumCorrect(test5)[2,2])/(NumCorrect(test5)[1,1]+NumCorrect(t
LDA_predictions = rbind(LDA_fold1_correct,LDA_fold2_correct,LDA_fold3_correct,LDA_fold4_correct,LDA_fol
LDA_predictions
```

```
##                    [,1]
## LDA_fold1_correct 1.000
## LDA_fold2_correct 1.000
## LDA_fold3_correct 1.000
## LDA_fold4_correct 1.000
## LDA_fold5_correct 0.975
```

Assumptions for logistic regression:

1. The outcome is binary (0 or 1). From the data, we know that the first 100 notes are genuine and the second 100 are counterfeit. This relates to a 0 and 1 respectively for the column "Counterfeit" that we added.

2. There is a linear relationship between the predictors and the outcome. This may simply be assumed, as the results will indicate whether this was correct.

3. There are no outliers. We could check for this fairly simply. One way would be to compute the column means/SD and find the minimum and maximum. We will do this for one and the rest will be assumed. As we will see, the max and min are extremely close to the mean.

```
mean(notes$Length)
```

```
## [1] 214.896
```

```
sd(notes$Length)
```

```
## [1] 0.3765541
```

```
max(notes$Length)
```

```
## [1] 216.3
```

```
min(notes$Length)
```

```
## [1] 213.8
```

We must create all.fits for each training set, as they will be used in the function later. I omitted the summaries for everything other than all.fit1, simply for ease of viewrship. The z-scores were similar. The code outputs an error, because it thinks the model is too efficient at prediciting.

```
library(ISLR)
  all.fit1 = glm(Counterfeit ~ Length + Left + Right + Bottom + Top +
  Diagonal, data = train1, family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge
```

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

```
  summary(all.fit1)
```

```
##
## Call:
## glm(formula = Counterfeit ~ Length + Left + Right + Bottom +
##     Top + Diagonal, family = binomial, data = train1)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -6.410e-05  -2.100e-08  -2.100e-08   2.100e-08   5.211e-05
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  7.627e+03  9.076e+06   0.001    0.999
```

```
## Length      -6.758e+00  3.980e+04   0.000    1.000
## Left         4.041e+01  7.015e+04   0.001    1.000
## Right       -5.060e+01  4.899e+04  -0.001    0.999
## Bottom       5.206e+01  2.339e+04   0.002    0.998
## Top          3.893e+01  2.398e+04   0.002    0.999
## Diagonal    -4.097e+01  1.711e+04  -0.002    0.998
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2.2171e+02  on 159  degrees of freedom
## Residual deviance: 1.3182e-08  on 153  degrees of freedom
## AIC: 14
##
## Number of Fisher Scoring iterations: 25
```

```
all.fit2 = glm(Counterfeit ~ Length + Left + Right + Bottom + Top +
Diagonal, data = train2, family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge

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

```
all.fit3 = glm(Counterfeit ~ Length + Left + Right + Bottom + Top +
Diagonal, data = train3, family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge

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

```
all.fit4 = glm(Counterfeit ~ Length + Left + Right + Bottom + Top +
Diagonal, data = train4, family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge

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

```
all.fit5 = glm(Counterfeit ~ Length + Left + Right + Bottom + Top +
Diagonal, data = train5, family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge

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

The next two r-blocks are the code for Logistic regression. We then put the prediction values into validation set in an additional column labeled "predictions". Once again we use functions for ease of viewership.

```
pred_all = function(obs,all.fit){
  x = c(1, obs)
  pred = as.numeric(as.numeric(x %*% all.fit$coefficients))
  pred = 1/(1+exp(-pred))
  return(pred)
}
```

```
LogReg = function(val,all.fit){
predictions = NULL
for(i in 1:40){
  predictions[i] = pred_all(as.matrix(val[i,1:6]),all.fit)
}
predictions
val$predictions = NULL
for(i in 1:40){
  val[i,8] = predictions[i]
}
val
}
```

Like above, this is a function for computing the number of correct predictions. Because the entries in the prediction column are not actually 0 or 1, we use > or < 0.5 to check correct predictions. We could even marginalize these numbers, and it would still show the same numbers. This is because the numbers are exceptionally close to 0 and 1.

```
NumberCorrect = function(test){
Correct1 = 0
Incorrect1 = 0
Correct0 = 0
Incorrect0 = 0
for(i in 1:40){
  if(test[i,7] == 1){
    if(test[i,8] > 0.5){
      Correct1 = Correct1 + 1
    }
    if(test[i,8] < 0.5){
      Incorrect1 = Incorrect1 + 1
    }
  }
  if(test[i,7] == 0){
    if(test[i,8] < 0.5){
      Correct0 = Correct0 + 1
    }
    if(test[i,8] > 0.5){
      Incorrect0 = Incorrect0 + 1
    }
  }
}
Table1 = rbind(Correct1 + Incorrect1,Correct1,Incorrect1)
Table0 = rbind(Correct0 + Incorrect0,Correct0,Incorrect0)
Table_full = cbind(Table1,Table0)
colnames(Table_full) = c("Counterfeit1","Counterfeit0")
rownames(Table_full) = c("Number of Observations","Correct","Incorrect")
Table_full
}
```

The next 5 r-blocks are performing Logistic Regression on the 5 different folds. As before, we compute the number of correct predictions for later use. Once again, we will only demonstrate the first prediction table.

```r
Test1 = LogReg(val1,all.fit1)
Test1
```

```
##      Length  Left Right Bottom  Top Diagonal Counterfeit           V8
## BN1   214.6 129.7 129.8    7.9 10.3    141.1           0 1.098586e-52
## BN2   215.0 130.0 129.8    8.6 10.6    141.5           0 8.153186e-35
## BN3   214.6 129.7 129.3   10.4 11.0    139.3           1 1.000000e+00
## BN4   214.4 129.9 129.6    7.5 10.5    141.8           0 2.599212e-62
## BN5   214.9 130.6 130.4   11.9 10.5    139.8           1 1.000000e+00
## BN6   215.1 130.6 130.3   12.3 10.2    139.6           1 1.000000e+00
## BN7   214.3 129.9 129.9   10.2 11.5    139.6           1 1.000000e+00
## BN8   215.5 130.7 130.3   10.2 11.8    140.0           1 1.000000e+00
## BN9   214.8 129.9 129.7    8.3 10.2    141.5           0 2.490421e-47
## BN10  215.7 130.2 130.0    8.7 10.0    141.6           0 2.037799e-47
## BN11  213.8 129.8 129.5    8.4 11.1    140.9           0 1.332577e-13
## BN12  214.7 130.0 129.4    7.8 10.0    141.2           0 4.910400e-48
## BN13  214.9 130.3 129.9    7.4 11.2    141.5           0 1.931375e-48
## BN14  214.3 130.3 130.0   11.4 10.5    139.8           1 1.000000e+00
## BN15  215.2 130.6 130.8   10.4 11.2    140.3           1 1.000000e+00
## BN16  214.7 130.2 129.9    8.6 10.0    141.9           0 6.960350e-50
## BN17  214.5 130.1 130.1   12.1 10.3    139.4           1 1.000000e+00
## BN18  214.9 130.4 129.9   11.4 11.0    139.9           1 1.000000e+00
## BN19  215.2 130.4 130.3    8.0 11.5    139.2           1 9.998836e-01
## BN20  214.3 130.2 130.0   10.7 10.5    139.8           1 1.000000e+00
## BN21  214.4 129.8 129.2    8.9  9.4    142.3           0 4.057185e-51
## BN22  214.8 130.0 129.7   11.4 10.6    139.2           1 1.000000e+00
## BN23  215.5 130.2 130.1    8.9  9.8    142.4           0 4.009297e-62
## BN24  214.7 130.1 130.2   11.6 10.9    139.1           1 1.000000e+00
## BN25  215.0 130.2 129.9   10.0 11.9    139.4           1 1.000000e+00
## BN26  215.0 129.6 129.7   10.4  7.7    141.8           0 2.619301e-53
## BN27  214.9 130.5 130.1    9.9 10.2    138.1           1 1.000000e+00
## BN28  215.4 130.0 129.9    8.5  9.7    141.4           0 6.965852e-54
## BN29  215.0 130.4 130.6    9.9 10.9    140.3           1 1.000000e+00
## BN30  214.8 129.7 129.7    8.7  9.6    142.2           0 2.136360e-64
## BN31  215.0 129.6 129.4    8.8  9.0    141.1           0 1.872289e-48
## BN32  214.8 130.1 129.6    8.8  9.9    140.9           0 1.030643e-24
## BN33  215.1 130.2 129.8   10.2 12.0    139.4           1 1.000000e+00
## BN34  214.4 130.1 130.0   11.3 10.7    139.2           1 1.000000e+00
## BN35  214.9 129.6 129.4    9.3  9.0    141.7           0 1.559787e-47
## BN36  214.6 130.2 130.4   11.2 10.7    139.9           1 1.000000e+00
## BN37  214.9 129.9 130.0    9.9 12.3    139.4           1 1.000000e+00
## BN38  214.9 130.0 129.9   11.4 10.9    139.7           1 1.000000e+00
## BN39  214.8 130.1 130.1   11.9 11.1    139.5           1 1.000000e+00
## BN40  214.6 130.2 130.2    9.4  9.7    141.8           0 2.173513e-41
```

```r
NumberCorrect(Test1)
```

```
##                        Counterfeit1 Counterfeit0
## Number of Observations           22           18
## Correct                          22           18
## Incorrect                         0            0
```

```r
LogReg_fold1_correct = (NumberCorrect(Test1)[2,1]+NumberCorrect(Test1)[2,2])/(NumberCorrect(Test1)[1,1]
```

```r
Test2 = LogReg(val2,all.fit2)
NumberCorrect(Test2)
```

```
##                        Counterfeit1 Counterfeit0
## Number of Observations           24           16
## Correct                          23           16
## Incorrect                         1            0
```

```r
LogReg_fold2_correct = (NumberCorrect(Test2)[2,1]+NumberCorrect(Test2)[2,2])/(NumberCorrect(Test2)[1,1]
```

```r
Test3 = LogReg(val3,all.fit3)
NumberCorrect(Test3)
```

```
##                        Counterfeit1 Counterfeit0
## Number of Observations           18           22
## Correct                          18           21
## Incorrect                         0            1
```

```r
LogReg_fold3_correct = (NumberCorrect(Test3)[2,1]+NumberCorrect(Test3)[2,2])/(NumberCorrect(Test3)[1,1]
```

```r
Test4 = LogReg(val4,all.fit4)
NumberCorrect(Test4)
```

```
##                        Counterfeit1 Counterfeit0
## Number of Observations           18           22
## Correct                          18           22
## Incorrect                         0            0
```

```r
LogReg_fold4_correct = (NumberCorrect(Test4)[2,1]+NumberCorrect(Test4)[2,2])/(NumberCorrect(Test4)[1,1]
```

```r
Test5 = LogReg(val5,all.fit5)
NumberCorrect(Test5)
```

```
##                        Counterfeit1 Counterfeit0
## Number of Observations           18           22
## Correct                          18           21
## Incorrect                         0            1
```

```r
LogReg_fold5_correct = (NumberCorrect(Test5)[2,1]+NumberCorrect(Test5)[2,2])/(NumberCorrect(Test5)[1,1]
LogReg_predictions = rbind(LogReg_fold1_correct,LogReg_fold2_correct,LogReg_fold3_correct,LogReg_fold4_
LogReg_predictions
```

```
##                       [,1]
## LogReg_fold1_correct 1.000
## LogReg_fold2_correct 0.975
## LogReg_fold3_correct 0.975
## LogReg_fold4_correct 1.000
## LogReg_fold5_correct 0.975
```

Assumptions for MLE:

LDA has extremely similar assumptions to LDA. It needs the data to be independently sampled, have common mean vector, and be approximately normally distributed. Look at the assumptions above for explanation, as it uses the same data.

Here, we perform MLE dimension reduction, and check which number of factors accounts for the majority of the variance. Because 2 factors accounts for the majority, we will use that on which to perform LDA and LogReg. 3 is also a very acceptable number to use. We notice that the first factor loads heavily on the bottom margin and somewhat the left width, right width, and diagonal. We then notice that the second variable loads heavily on the top margin and again somewhat on the left margin, right margin, and diagonal. It seems the top and bottom margins account for quite a lot of the variance on the notes. This would make sense, as the width of the left, right, diagnoal, and total would all be geometrically related, so they would just be sprinkled in with the other two main factors.

```
n.factors = 3

fa_fit = factanal(notes[,1:6], factors = n.factors, rotation = "varimax")
fa_fit
```

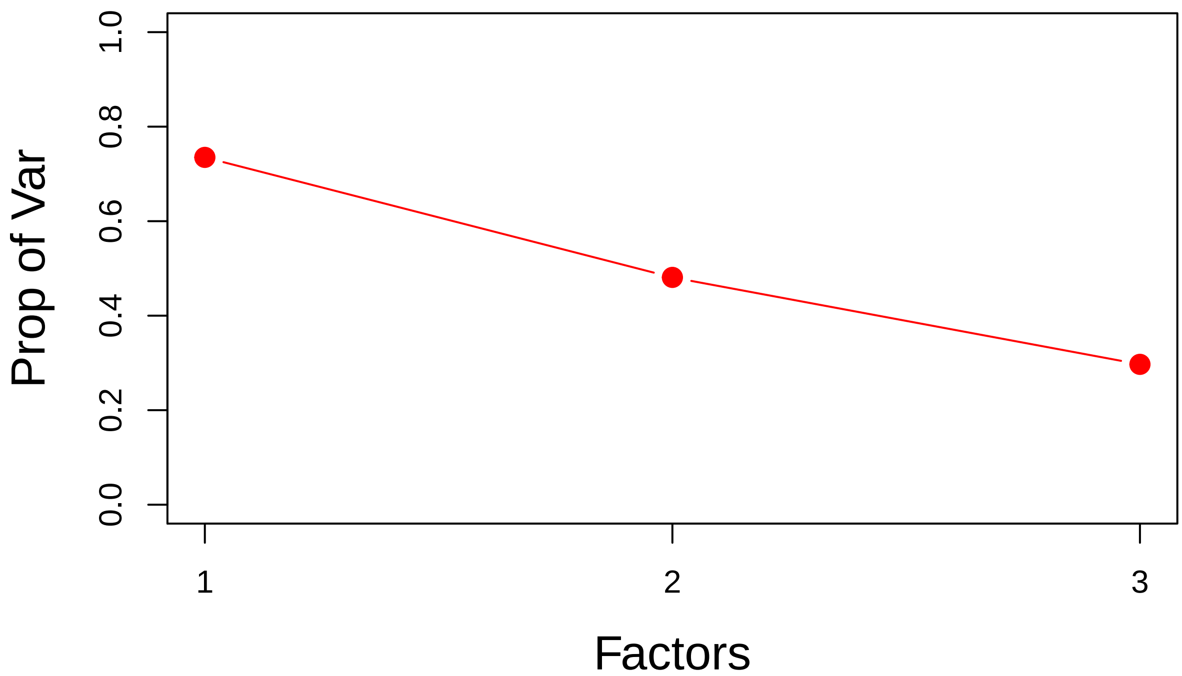
```
##
## Call:
## factanal(x = notes[, 1:6], factors = n.factors, rotation = "varimax")
##
## Uniquenesses:
##   Length     Left    Right   Bottom      Top Diagonal
##    0.716    0.205    0.292    0.005    0.402    0.164
##
## Loadings:
##          Factor1 Factor2 Factor3
## Length   -0.180  -0.132   0.484
## Left      0.351   0.410   0.710
## Right     0.427   0.406   0.601
## Bottom    0.987   0.142
## Top               0.769
## Diagonal -0.523  -0.750
##
##                Factor1 Factor2 Factor3
## SS loadings      1.587   1.524   1.106
## Proportion Var   0.265   0.254   0.184
## Cumulative Var   0.265   0.519   0.703
##
## The degrees of freedom for the model is 0 and the fit was 0.0189
```

```
loading = fa_fit$loadings[,1:3]
weights = t(loading)
weights
```

```
##             Length      Left     Right     Bottom        Top     Diagonal
## Factor1 -0.1798542 0.3509698 0.4266324 0.98730708 0.03239197 -0.523363912
## Factor2 -0.1316173 0.4103754 0.4058786 0.14161289 0.76908314 -0.749832691
## Factor3  0.4840111 0.7096473 0.6014242 0.01307913 0.07635236 -0.004236968
```

Here, we simply graph a scree plot to show the proportion of variance accounted for with the number of factors. As said above, because 2 factors accounts for more than half, we will choose that.

```
plot(x = c(1,2,3),y = c(1-0.265,1-0.519,1-0.703), xlab = "Factors", ylab = "Prop of Var", ylim = c(0,1)
     type = "b", col = "red", cex = 2, pch = 20, cex.lab = 1.5)
axis(1, at = c(1,2,3), labels = c(1,2,3))
```



Before I realized that we were supposed to use only factor analysis, I had conducted PCA. Ignore this r-block, however, I suppose I'll leave it in, just to demonstrate ability.

```
pca_result = prcomp(notes[,1:6],scale = TRUE)
pca_var = pca_result$sdev^2
pve = pca_var/sum(pca_var)
output = cbind(pca_var, pve, cumsum(pve))
colnames(output) = c("Eigenval", "Proportion", "Cumulative")
rownames(output) = c("PC1", "PC2", "PC3", "PC4", "PC5", "PC6")
output
```

```
##      Eigenval Proportion Cumulative
## PC1 2.9455582 0.49092637  0.4909264
## PC2 1.2780838 0.21301396  0.7039403
## PC3 0.8690326 0.14483876  0.8487791
## PC4 0.4497687 0.07496145  0.9237405
## PC5 0.2686769 0.04477948  0.9685200
## PC6 0.1888799 0.03147998  1.0000000
```

```r
t(pca_result$rotation)
```

```
##          Length       Left      Right     Bottom         Top   Diagonal
## PC1  0.006987029 -0.4677582 -0.4866787 -0.4067583 -0.36789112  0.4934583
## PC2 -0.815494969 -0.3419671 -0.2524586  0.2662288  0.09148667 -0.2739407
## PC3 -0.017680661  0.1033829  0.1234747  0.5835383 -0.78757147  0.1138754
## PC4 -0.574617276  0.3949225  0.4302783 -0.4036735 -0.11022672  0.3919305
## PC5  0.058796102 -0.6394961  0.6140972  0.2154756  0.21984942  0.3401601
## PC6  0.031056981 -0.2977477  0.3491529 -0.4623536 -0.41896754 -0.6317985
```

Now, we create a new 200x2 matrix with the linear combinations of loadings(weights) with the note rows creating 2 factors. We will also leave in the Counterfeit binary column.

```r
FAC = notes[200,1:3]
for(i in 1:200){
  FAC[i,1] = weights[1,1]*notes[i,1] + weights[1,2]*notes[i,2] + weights[1,3]*notes[i,3] + weights[1,4]
  FAC[i,2] = weights[2,1]*notes[i,1] + weights[2,2]*notes[i,2] + weights[2,3]*notes[i,3] + weights[2,4]
  FAC[i,3] = notes[i,7]
}
colnames(FAC) = c("FAC1", "FAC2", "Counterfeit")
FAC
```

```
##              FAC1      FAC2 Counterfeit
## BN200 -3.41232169 -19.09745           0
## 2     -2.88748543 -18.99706           0
## 3     -0.19264076 -17.05830           1
## 4     -4.14628255 -19.49794           0
## 5      1.74165411 -16.82902           1
## 6      2.15289806 -16.92004           1
## 7     -0.15078573 -16.56194           1
## 8     -0.11480990 -16.29844           1
## 9     -3.23842373 -19.40248           0
## 10    -2.83090378 -19.44822           0
## 11    -2.73709119 -18.23684           0
## 12    -3.65645381 -19.46972           0
## 13    -3.88687915 -18.52869           0
## 14     1.07996917 -17.10632           1
## 15     0.13838261 -16.75513           1
## 16    -2.94945273 -19.59630           0
## 17     1.91044974 -16.92889           1
## 18     0.92835001 -16.87528           1
## 19    -1.92924663 -16.32448           1
## 20     0.35375724 -17.24649           1
## 21    -3.26711570 -20.72398           0
## 22     1.07401898 -16.89020           1
## 23    -2.97997782 -20.10667           0
## 24     1.59993296 -16.29903           1
## 25    -0.25122454 -16.10169           1
## 26    -1.54432975 -21.40219           0
## 27     0.48395432 -16.23106           1
## 28    -2.99231095 -19.64048           0
## 29    -0.48453812 -17.19359           1
```

```
## 30    -3.29948479 -20.41425           0
## 31    -2.84354648 -20.22584           0
## 32    -2.41293871 -19.07101           0
## 33    -0.11117258 -16.05020           1
## 34     1.21355583 -16.61200           1
## 35    -2.64592588 -20.59177           0
## 36     0.91824948 -16.97398           1
## 37    -0.38164075 -15.87758           1
## 38     0.88939567 -16.96637           1
## 39     1.63260926 -16.45641           1
## 40    -1.97101314 -19.50382           0
## 41    -2.71209936 -20.07198           0
## 42    -0.20730273 -15.86757           1
## 43     2.01130319 -16.23859           1
## 44     1.79249416 -16.61171           1
## 45    -4.39633016 -19.85954           0
## 46    -2.42477410 -19.24705           0
## 47     0.27419537 -16.01838           1
## 48     0.96284036 -16.06534           1
## 49    -2.00290160 -18.72969           0
## 50     0.36743848 -16.07715           1
## 51    -0.31531978 -16.00118           1
## 52    -1.92344877 -16.68813           1
## 53    -0.08099437 -14.80019           1
## 54     1.72132709 -17.13010           1
## 55    -3.59745544 -20.17488           0
## 56     0.92205543 -16.08465           1
## 57    -0.02519030 -15.53366           1
## 58    -3.34379434 -19.66689           0
## 59     1.02452091 -17.10424           1
## 60    -3.44593635 -18.68273           0
## 61    -1.82148440 -16.90409           1
## 62     0.15947893 -16.79911           1
## 63    -2.62091571 -19.22120           0
## 64    -4.16806146 -19.46037           0
## 65    -3.71405675 -19.31966           0
## 66    -2.47511127 -18.75273           0
## 67    -2.29699689 -20.60426           0
## 68    -0.51977216 -16.39764           1
## 69    -3.22878153 -20.01459           0
## 70     1.14906017 -16.87519           1
## 71     1.30223626 -17.04450           1
## 72    -3.85909388 -19.15874           0
## 73    -0.20252269 -16.01254           1
## 74    -0.56319047 -16.48128           1
## 75    -2.83149554 -20.55251           0
## 76     2.14884398 -16.13915           1
## 77     1.35767916 -16.65002           1
## 78     2.06123792 -16.80349           1
## 79    -3.79434370 -19.78484           0
## 80    -1.54230931 -15.29087           1
## 81    -3.30045881 -19.65583           0
## 82    -0.58747777 -16.13336           1
## 83    -2.44112127 -19.09545           0
```

```
## 84     0.66216290 -16.83095           1
## 85     1.68939680 -16.99351           1
## 86    -0.85956431 -16.36877           1
## 87    -0.05618749 -16.51443           1
## 88    -3.23686468 -20.12770           0
## 89    -3.36030706 -18.64659           0
## 90    -3.14324462 -18.24171           0
## 91    -3.74850607 -18.79116           0
## 92    -3.17096538 -20.36608           0
## 93    -3.46674787 -20.88793           0
## 94    -3.88885301 -19.93333           0
## 95    -3.25777488 -19.22261           0
## 96    -4.16516124 -19.40084           0
## 97    -2.59211510 -18.72980           0
## 98     0.37217130 -16.68085           1
## 99    -1.97496729 -17.93396           0
## 100    0.20695183 -15.89525           1
## 101    1.21479804 -16.91186           1
## 102   -3.43884699 -19.39858           0
## 103    1.56113625 -16.63898           1
## 104    1.38607348 -16.87712           1
## 105   -4.11942371 -20.17543           0
## 106   -1.31847069 -18.29334           0
## 107    0.21980665 -16.57693           1
## 108   -3.97947722 -18.98386           0
## 109   -0.20779702 -16.86110           1
## 110    0.55002036 -16.57407           1
## 111   -1.84373901 -18.56687           0
## 112    1.42628617 -17.07816           1
## 113    0.31778703 -16.44123           1
## 114    0.20829416 -17.01920           1
## 115   -2.99045334 -20.01183           0
## 116    1.40466461 -16.43813           1
## 117   -3.45794626 -19.25371           0
## 118   -3.93704309 -18.80894           0
## 119   -3.79265826 -19.37911           0
## 120   -0.50913467 -16.62433           1
## 121   -3.96501493 -19.16797           0
## 122    1.65702317 -17.06868           1
## 123   -0.71446274 -16.00935           1
## 124   -3.80828934 -20.19060           0
## 125   -4.30666256 -19.55470           0
## 126   -3.17041846 -19.80403           0
## 127   -3.73324422 -19.22952           0
## 128   -2.67251735 -18.67425           0
## 129   -3.11535038 -18.23670           0
## 130   -2.73321280 -20.01580           0
## 131    0.83847666 -16.19526           1
## 132   -4.02233822 -19.29044           0
## 133   -0.98850736 -16.73223           1
## 134   -2.74265897 -19.39759           0
## 135    0.16859709 -17.19277           1
## 136   -4.42445254 -20.02583           0
## 137   -3.45911753 -18.90919           0
```

```
## 138   -3.54319292 -19.00607          0
## 139   -0.89099095 -14.96502          1
## 140   -3.67921608 -20.12234          0
## 141    2.61264225 -16.74142          1
## 142   -2.90756824 -20.52170          0
## 143    0.83442327 -16.25141          1
## 144   -1.03792767 -14.79421          1
## 145    1.14274974 -16.72585          1
## 146   -1.64707196 -19.66016          0
## 147   -2.44157188 -19.11135          0
## 148   -1.28808503 -16.56414          1
## 149    1.02867293 -17.42863          1
## 150   -2.06373175 -18.14278          0
## 151   -0.10788682 -16.56391          1
## 152    1.53625058 -16.78099          1
## 153   -0.24860456 -16.00043          1
## 154   -1.57811196 -18.43291          0
## 155   -3.82255105 -19.61548          0
## 156   -4.69746474 -19.75317          0
## 157   -3.62566883 -19.94689          0
## 158   -0.18315608 -16.74038          1
## 159    0.86275084 -16.42843          1
## 160   -1.40067853 -15.76998          1
## 161    0.47352029 -16.60756          1
## 162   -3.57388346 -19.84754          0
## 163    0.00773563 -15.83803          1
## 164   -3.50693681 -19.49044          0
## 165   -4.04062928 -19.60134          0
## 166   -0.53295224 -17.11522          1
## 167   -4.31129584 -20.52628          0
## 168    0.07167126 -15.89019          1
## 169   -3.24816023 -19.68557          0
## 170   -4.46473839 -19.38065          0
## 171   -1.57676416 -18.97394          0
## 172    0.50092317 -16.57215          1
## 173    1.20858242 -17.10966          1
## 174   -3.42565983 -18.73711          0
## 175   -3.40023053 -18.14755          0
## 176    2.66350819 -17.50735          1
## 177    0.10670566 -16.30511          1
## 178   -3.40119602 -18.57013          0
## 179   -4.36788874 -19.43241          0
## 180   -2.83651634 -18.58365          0
## 181   -3.77982297 -20.94473          0
## 182   -3.63037786 -20.03314          0
## 183    0.06333064 -15.78168          1
## 184   -4.30478639 -18.74647          0
## 185    1.52427986 -15.86015          1
## 186   -4.35338299 -19.69858          0
## 187   -4.67758920 -19.61025          0
## 188   -0.02431697 -15.40047          1
## 189    0.43356112 -16.39922          1
## 190    0.28561178 -16.81158          1
## 191   -3.29029407 -19.16619          0
```

```
## 192    0.42163917 -15.38281           1
## 193   -3.97140398 -20.25184           0
## 194   -2.20721074 -16.93831           0
## 195   -0.35291053 -17.46514           1
## 196   -2.51347740 -15.69051           1
## 197    1.55090452 -16.93862           1
## 198   -3.16301965 -20.25554           0
## 199   -2.08874773 -18.96906           0
## 200    0.03369578 -15.81337           1
```

Once again, we separate bank notes into 5 folds now using the factors as variables. This separation will have the identical bank notes in the folds.

```
FAC_fold1 = FAC[1:40,]
FAC_fold2 = FAC[41:80,]
FAC_fold3 = FAC[81:120,]
FAC_fold4 = FAC[121:160,]
FAC_fold5 = FAC[161:200,]
```

Unfortunately, because the dimensions are different, we have to rewrite the function for LDA with the new dimensions for the factors. It is basically the same function as before. You may ignore this.

```
FAC_LDA = function(train,val){
  Counter1_TR = train[train$Counterfeit == "1",1:2]
  Counter0_TR = train[train$Counterfeit == "0",1:2]
  n1_TR = dim(Counter1_TR)[1]
  n0_TR = dim(Counter0_TR)[1]
  N_TR = n1_TR + n0_TR
  Counter1_p = n1_TR/N_TR
  Counter0_p = n0_TR/N_TR
  Counter1_mu = colMeans(Counter1_TR)
  Counter0_mu = colMeans(Counter0_TR)
  rbind(Counter1_mu,Counter0_mu)
  Counter1_S = cov(Counter1_TR)
  Counter0_S = cov(Counter0_TR)
  S_pool = ((n1_TR-1)*Counter1_S + (n0_TR-1)*Counter0_S)/(n1_TR + n0_TR - 1)
  S_inv = solve(S_pool)
  Counter1_alpha = -0.5*t(Counter1_mu) %*% S_inv %*% Counter1_mu +    log(Counter1_p)
  Counter0_alpha = -0.5*t(Counter0_mu) %*% S_inv %*% Counter0_mu + log(Counter0_p)
  Counter01_alpha = c(Counter1_alpha,Counter0_alpha)
  Counter01_alpha
  Counter1_beta = S_inv %*% Counter1_mu
  Counter0_beta = S_inv %*% Counter0_mu
  Counter01_beta = cbind(Counter1_beta,Counter0_beta)
  Counter01_beta

  prediction = c()
Counter1_dvec = c()
Counter0_dvec = c()
label = c("1","0")
for(i in 1:nrow(val)){
x = t(val[i,1:2])
Counter1_d = Counter1_alpha + t(Counter1_beta) %*% x
```

```
Counter0_d = Counter0_alpha + t(Counter0_beta) %*% x
dvec = c(Counter1_d,Counter0_d)
prediction = append(prediction, label[which.max(dvec)])
Counter1_dvec = append(Counter1_dvec,Counter1_d)
Counter0_dvec = append(Counter0_dvec,Counter0_d)
}
val$prediction = prediction
val
}
```

Same thing as above, same function, different dimensions. You may ignore.

```
FAC_NumCorrect = function(test){
Correct1 = 0
Incorrect1 = 0
Correct0 = 0
Incorrect0 = 0
for(i in 1:40){
  if(test[i,3] == 1){
    if(test[i,3] == test[i,4]){
      Correct1 = Correct1 + 1
    }
    if(test[i,3] != test[i,4]){
      Incorrect1 = Incorrect1 + 1
    }
  }
  if(test[i,3] == 0){
    if(test[i,3] == test[i,4]){
      Correct0 = Correct0 + 1
    }
    if(test[i,3] != test[i,4]){
      Incorrect0 = Incorrect0 + 1
    }
  }
}
Table1 = rbind(Correct1 + Incorrect1,Correct1,Incorrect1)
Table0 = rbind(Correct0 + Incorrect0,Correct0,Incorrect0)
Table_full = cbind(Table1,Table0)
colnames(Table_full) = c("Counterfeit1","Counterfeit0")
rownames(Table_full) = c("Number of Observations","Correct","Incorrect")
Table_full
}
```

We now, like above, conduct LDA on the 5 folds, now using the 2 factors. We will later see if this changed the prediction accuracy.

```
FAC_val1 = FAC_fold1
FAC_train1 = rbind(FAC_fold2,FAC_fold3,FAC_fold4,FAC_fold5)
FAC_test1 = FAC_LDA(FAC_train1,FAC_val1)
FAC_test1
```

```
##             FAC1      FAC2 Counterfeit prediction
## BN200 -3.4123217 -19.09745           0          0
```

```
## 2     -2.8874854 -18.99706          0          0
## 3     -0.1926408 -17.05830          1          1
## 4     -4.1462825 -19.49794          0          0
## 5      1.7416541 -16.82902          1          1
## 6      2.1528981 -16.92004          1          1
## 7     -0.1507857 -16.56194          1          1
## 8     -0.1148099 -16.29844          1          1
## 9     -3.2384237 -19.40248          0          0
## 10    -2.8309038 -19.44822          0          0
## 11    -2.7370912 -18.23684          0          0
## 12    -3.6564538 -19.46972          0          0
## 13    -3.8868791 -18.52869          0          0
## 14     1.0799692 -17.10632          1          1
## 15     0.1383826 -16.75513          1          1
## 16    -2.9494527 -19.59630          0          0
## 17     1.9104497 -16.92889          1          1
## 18     0.9283500 -16.87528          1          1
## 19    -1.9292466 -16.32448          1          1
## 20     0.3537572 -17.24649          1          1
## 21    -3.2671157 -20.72398          0          0
## 22     1.0740190 -16.89020          1          1
## 23    -2.9799778 -20.10667          0          0
## 24     1.5999330 -16.29903          1          1
## 25    -0.2512245 -16.10169          1          1
## 26    -1.5443297 -21.40219          0          0
## 27     0.4839543 -16.23106          1          1
## 28    -2.9923110 -19.64048          0          0
## 29    -0.4845381 -17.19359          1          1
## 30    -3.2994848 -20.41425          0          0
## 31    -2.8435465 -20.22584          0          0
## 32    -2.4129387 -19.07101          0          0
## 33    -0.1111726 -16.05020          1          1
## 34     1.2135558 -16.61200          1          1
## 35    -2.6459259 -20.59177          0          0
## 36     0.9182495 -16.97398          1          1
## 37    -0.3816408 -15.87758          1          1
## 38     0.8893957 -16.96637          1          1
## 39     1.6326093 -16.45641          1          1
## 40    -1.9710131 -19.50382          0          0
```

```r
FAC_NumCorrect(FAC_test1)
```

```
##                        Counterfeit1 Counterfeit0
## Number of Observations           22           18
## Correct                          22           18
## Incorrect                         0            0
```

```r
FAC_LDA_fold1_correct = (FAC_NumCorrect(FAC_test1)[2,1]+FAC_NumCorrect(FAC_test1)[2,2])/(FAC_NumCorrect
```

```r
FAC_val2 = FAC_fold2
FAC_train2 = rbind(FAC_fold1,FAC_fold3,FAC_fold4,FAC_fold5)
FAC_test2 = FAC_LDA(FAC_train2,FAC_val2)
FAC_NumCorrect(FAC_test2)
```

```
##                          Counterfeit1 Counterfeit0
## Number of Observations            24           16
## Correct                           24           16
## Incorrect                          0            0
```

```
FAC_LDA_fold2_correct = (FAC_NumCorrect(FAC_test2)[2,1]+FAC_NumCorrect(FAC_test2)[2,2])/(FAC_NumCorrect
```

```
FAC_val3 = FAC_fold3
FAC_train3 = rbind(FAC_fold1,FAC_fold2,FAC_fold4,FAC_fold5)
FAC_test3 = FAC_LDA(FAC_train3,FAC_val3)
FAC_NumCorrect(FAC_test3)
```

```
##                          Counterfeit1 Counterfeit0
## Number of Observations            18           22
## Correct                           18           22
## Incorrect                          0            0
```

```
FAC_LDA_fold3_correct = (FAC_NumCorrect(FAC_test3)[2,1]+FAC_NumCorrect(FAC_test3)[2,2])/(FAC_NumCorrect
```

```
FAC_val4 = FAC_fold4
FAC_train4 = rbind(FAC_fold1,FAC_fold2,FAC_fold3,FAC_fold5)
FAC_test4 = FAC_LDA(FAC_train4,FAC_val4)
FAC_NumCorrect(FAC_test4)
```

```
##                          Counterfeit1 Counterfeit0
## Number of Observations            18           22
## Correct                           18           22
## Incorrect                          0            0
```

```
FAC_LDA_fold4_correct = (FAC_NumCorrect(FAC_test4)[2,1]+FAC_NumCorrect(FAC_test4)[2,2])/(FAC_NumCorrect
```

```
FAC_val5 = FAC_fold5
FAC_train5 = rbind(FAC_fold1,FAC_fold2,FAC_fold3,FAC_fold4)
FAC_test5 = FAC_LDA(FAC_train5,FAC_val5)
FAC_NumCorrect(FAC_test5)
```

```
##                          Counterfeit1 Counterfeit0
## Number of Observations            18           22
## Correct                           18           21
## Incorrect                          0            1
```

```
FAC_LDA_fold5_correct = (FAC_NumCorrect(FAC_test5)[2,1]+FAC_NumCorrect(FAC_test5)[2,2])/(FAC_NumCorrect
FAC_LDA_predictions = rbind(FAC_LDA_fold1_correct,FAC_LDA_fold2_correct,FAC_LDA_fold3_correct,FAC_LDA_f
FAC_LDA_predictions
```

```
##                        [,1]
## FAC_LDA_fold1_correct 1.000
## FAC_LDA_fold2_correct 1.000
## FAC_LDA_fold3_correct 1.000
## FAC_LDA_fold4_correct 1.000
## FAC_LDA_fold5_correct 0.975
```

Like the first Logistic Regression, we must create all.fits for each of the training sets now comprising of the 2 factors. We omit the other summaries. Once again, the z-scores are extremely close to 0. Ignore the warning message again.

```
FAC_all.fit1 = glm(Counterfeit ~ FAC1 + FAC2, data = FAC_train1, family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge

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

```
summary(FAC_all.fit1)
```

```
##
## Call:
## glm(formula = Counterfeit ~ FAC1 + FAC2, family = binomial, data = FAC_train1)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -1.438e-04  -2.100e-08  -2.100e-08   2.100e-08   1.680e-04
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1196.13  284991.73   0.004    0.997
## FAC1            89.21   19480.28   0.005    0.996
## FAC2            60.08   15433.70   0.004    0.997
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2.2171e+02  on 159  degrees of freedom
## Residual deviance: 5.1922e-08  on 157  degrees of freedom
## AIC: 6
##
## Number of Fisher Scoring iterations: 25
```

```
FAC_all.fit2 = glm(Counterfeit ~ FAC1 + FAC2, data = FAC_train2, family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge

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

```
FAC_all.fit3 = glm(Counterfeit ~ FAC1 + FAC2, data = FAC_train3, family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge

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

```
FAC_all.fit4 = glm(Counterfeit ~ FAC1 + FAC2, data = FAC_train4, family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge

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

```
  FAC_all.fit5 = glm(Counterfeit ~ FAC1 + FAC2, data = FAC_train5, family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge

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

We again create the Logistic Regression function and number of correct predictions function with the new dimensions.

```
FAC_pred_all = function(obs,all.fit){
  x = c(1, obs)
  pred = as.numeric(as.numeric(x %*% all.fit$coefficients))
  pred = 1/(1+exp(-pred))
  return(pred)
}
```

```
FAC_LogReg = function(val,all.fit){
predictions = NULL
for(i in 1:40){
  predictions[i] = FAC_pred_all(as.matrix(val[i,1:2]),all.fit)
}
predictions
val$predictions = NULL
for(i in 1:40){
  val[i,4] = predictions[i]
}
val
}
```

```
FAC_NumberCorrect = function(test){
Correct1 = 0
Incorrect1 = 0
Correct0 = 0
Incorrect0 = 0
for(i in 1:40){
  if(test[i,3] == 1){
    if(test[i,4] > 0.5){
      Correct1 = Correct1 + 1
    }
    if(test[i,4] < 0.5){
      Incorrect1 = Incorrect1 + 1
    }
  }
  if(test[i,3] == 0){
    if(test[i,4] < 0.5){
      Correct0 = Correct0 + 1
    }
    if(test[i,4] > 0.5){
      Incorrect0 = Incorrect0 + 1
    }
  }
}
```

```
Table1 = rbind(Correct1 + Incorrect1,Correct1,Incorrect1)
Table0 = rbind(Correct0 + Incorrect0,Correct0,Incorrect0)
Table_full = cbind(Table1,Table0)
colnames(Table_full) = c("Counterfeit1","Counterfeit0")
rownames(Table_full) = c("Number of Observations","Correct","Incorrect")
Table_full
}
```

We conduct Logistic Regression on each of the 5 folds.

```
FAC_Test1 = FAC_LogReg(FAC_val1,FAC_all.fit1)
FAC_Test1
```

```
##             FAC1      FAC2 Counterfeit           V4
## BN200 -3.4123217 -19.09745           0 9.799018e-112
## 2     -2.8874854 -18.99706           0  8.775060e-89
## 3     -0.1926408 -17.05830           1  1.000000e+00
## 4     -4.1462825 -19.49794           0 1.279975e-150
## 5      1.7416541 -16.82902           1  1.000000e+00
## 6      2.1528981 -16.92004           1  1.000000e+00
## 7     -0.1507857 -16.56194           1  1.000000e+00
## 8     -0.1148099 -16.29844           1  1.000000e+00
## 9     -3.2384237 -19.40248           0 5.880285e-113
## 10    -2.8309038 -19.44822           0  2.311802e-98
## 11    -2.7370912 -18.23684           0  4.028519e-63
## 12    -3.6564538 -19.46972           0 6.607677e-131
## 13    -3.8868791 -18.52869           0 2.793941e-115
## 14     1.0799692 -17.10632           1  1.000000e+00
## 15     0.1383826 -16.75513           1  1.000000e+00
## 16    -2.9494527 -19.59630           0 8.081990e-107
## 17     1.9104497 -16.92889           1  1.000000e+00
## 18     0.9283500 -16.87528           1  1.000000e+00
## 19    -1.9292466 -16.32448           1  1.000000e+00
## 20     0.3537572 -17.24649           1  1.000000e+00
## 21    -3.2671157 -20.72398           0 1.505985e-148
## 22     1.0740190 -16.89020           1  1.000000e+00
## 23    -2.9799778 -20.10667           0 2.562129e-121
## 24     1.5999330 -16.29903           1  1.000000e+00
## 25    -0.2512245 -16.10169           1  1.000000e+00
## 26    -1.5443297 -21.40219           0  1.678889e-99
## 27     0.4839543 -16.23106           1  1.000000e+00
## 28    -2.9923110 -19.64048           0 1.243033e-109
## 29    -0.4845381 -17.19359           1  1.000000e+00
## 30    -3.2994848 -20.41425           0 1.012121e-141
## 31    -2.8435465 -20.22584           0 3.842355e-119
## 32    -2.4129387 -19.07101           0  2.501993e-72
## 33    -0.1111726 -16.05020           1  1.000000e+00
## 34     1.2135558 -16.61200           1  1.000000e+00
## 35    -2.6459259 -20.59177           0 4.930635e-121
## 36     0.9182495 -16.97398           1  1.000000e+00
## 37    -0.3816408 -15.87758           1  1.000000e+00
## 38     0.8893957 -16.96637           1  1.000000e+00
## 39     1.6326093 -16.45641           1  1.000000e+00
```

```
## 40    -1.9710131 -19.50382          0  1.683813e-66
```

```r
FAC_NumberCorrect(FAC_Test1)
```

```
##                          Counterfeit1 Counterfeit0
## Number of Observations           22           18
## Correct                          22           18
## Incorrect                         0            0
```

```r
FAC_LogReg_fold1_correct = (FAC_NumberCorrect(FAC_Test1)[2,1]+FAC_NumberCorrect(FAC_Test1)[2,2])/(FAC_Nu
```

```r
FAC_Test2 = FAC_LogReg(FAC_val2,FAC_all.fit2)
FAC_NumberCorrect(FAC_Test2)
```

```
##                          Counterfeit1 Counterfeit0
## Number of Observations           24           16
## Correct                          23           16
## Incorrect                         1            0
```

```r
FAC_LogReg_fold2_correct = (FAC_NumberCorrect(FAC_Test2)[2,1]+FAC_NumberCorrect(FAC_Test2)[2,2])/(FAC_Nu
```

```r
FAC_Test3 = FAC_LogReg(FAC_val3,FAC_all.fit3)
FAC_NumberCorrect(FAC_Test3)
```

```
##                          Counterfeit1 Counterfeit0
## Number of Observations           18           22
## Correct                          18           22
## Incorrect                         0            0
```

```r
FAC_LogReg_fold3_correct = (FAC_NumberCorrect(FAC_Test3)[2,1]+FAC_NumberCorrect(FAC_Test3)[2,2])/(FAC_Nu
```

```r
FAC_Test4 = FAC_LogReg(FAC_val4,FAC_all.fit4)
FAC_NumberCorrect(FAC_Test4)
```

```
##                          Counterfeit1 Counterfeit0
## Number of Observations           18           22
## Correct                          18           22
## Incorrect                         0            0
```

```r
FAC_LogReg_fold4_correct = (FAC_NumberCorrect(FAC_Test4)[2,1]+FAC_NumberCorrect(FAC_Test4)[2,2])/(FAC_Nu
```

```r
FAC_Test5 = FAC_LogReg(FAC_val5,FAC_all.fit5)
FAC_NumberCorrect(FAC_Test5)
```

```
##                          Counterfeit1 Counterfeit0
## Number of Observations           18           22
## Correct                          18           21
## Incorrect                         0            1
```

```r
FAC_LogReg_fold5_correct = (FAC_NumberCorrect(FAC_Test5)[2,1]+FAC_NumberCorrect(FAC_Test5)[2,2])/(FAC_Nu
FAC_LogReg_predictions = rbind(FAC_LogReg_fold1_correct,FAC_LogReg_fold2_correct,FAC_LogReg_fold3_corre
FAC_LogReg_predictions
```

```
##                           [,1]
## FAC_LogReg_fold1_correct 1.000
## FAC_LogReg_fold2_correct 0.975
## FAC_LogReg_fold3_correct 1.000
## FAC_LogReg_fold4_correct 1.000
## FAC_LogReg_fold5_correct 0.975
```

Now, we put all of our findings into 1 concise table. We will go into more depth in the conclusion, but it seems from the table that LDA was better at predictions both for the original variables and the new factors. Of course, we're comparing the difference between getting 1 wrong out of 200 and 2 (or 3) wrong out of 200. All models did exceptionally well at predictions. Therefore, it will probably come down to the other factors we mentioned before in determining the best model.

```r
Full_Predictions = matrix(,nrow = 5, ncol = 4)
colnames(Full_Predictions) = c("LDA","LogReg","FAC_LDA","FAC_LogReg")
rownames(Full_Predictions) = c("fold1","fold2","fold3","fold4","fold5")

Full_Predictions[,1] = LDA_predictions
Full_Predictions[,2] = LogReg_predictions
Full_Predictions[,3] = FAC_LDA_predictions
Full_Predictions[,4] = FAC_LogReg_predictions
Full_Predictions
```

```
##         LDA LogReg FAC_LDA FAC_LogReg
## fold1 1.000  1.000   1.000      1.000
## fold2 1.000  0.975   1.000      0.975
## fold3 1.000  0.975   1.000      1.000
## fold4 1.000  1.000   1.000      1.000
## fold5 0.975  0.975   0.975      0.975
```

Conclusion

In this project, we were trying to determine if we could use variables of length to verify the legitimacy of Swiss bank notes. We first visualized the data with histograms and a correlation plot which gave us a first look at relationships and distribution. The rest of the project was conducting LDA and Logistic Regression on each fold of variables and then on the factors we derived with factor analysis to reduce redundancy. Quite obviously, from the results in the table above, it was very possible to predict legitimacy based on these variables and the factors. To be honest, the factor analysis does indeed seem like a waste of time. We were already quite accurate with just predicting using all 6 variables. It does save some computation time, and increases accuracy by 1 result, but these seem negligible in the grand scheme. Because of the extra work it took to compute the factors and then apply it to the notes, we will disregard LDA and Logistic Regression on the factors as viable models. Before choosing between LDA and Logistic Regression, let's take a second to determine which fold was the best. Take into account that when it says fold(x) that means that fold was the validation set. In essence the other 4 were the training set. We must arrive at a final model for each fold. Because the prediction results were so close, we shall simply pick the best analyzer between LDA and Logistic Regression. Yes, LDA was more accurate, and you can see each part of the function. However, Logistic Regression has much more mild and easily calculated assumptions (binary, linear relationship, no outliers) compared with LDA (normality, common mean vector, homoskedacity, independently sampled). In addition, the process is extremely simple, and the results were almost equivalent to LDA (98.5 vs 99.5 accuracy). Therefore, we will choose Logistic Regression as the model for all 5 folds.