Part 4: Dimensionality Reduction

In this section, we'll perform dimensionality reduction with principal components analysis (PCA) and Linear Discriminant Analysis (LDA) on the 'Wine Rating and Price' dataset which can be found here. (Check out the classification section of this assignment to find out more about the wine data and what we will try to do with dimensionality reduction, as we'll make comparisons with the findings in that section.)

Wine Dataset

For PDF viewers: https://www.kaggle.com/datasets/budnyak/wine-rating-and-price?select=Red.csv

The wine data that was downloaded from kaggle was concatenated into a larger, total wine csv file. While there are distinct qualities in each type of wine that make them unique, we needed to concatenate them so that we have a large enough data set to perform analysis on. Additionally, all the wine categories held the same features, so a concatenation is not too messy to deal with.

```
wine <- read.csv("totalWine.csv", header = TRUE)
head(wine)
```

```
##                                Name Country    Region                 Winery
## 1                      Pomerol 2011  France   Pomerol Château La Providence
## 2                        Lirac 2017  France     Lirac    Château Mont-Redon
## 3 Erta e China Rosso di Toscana 2015   Italy   Toscana            Renzo Masi
## 4                    Bardolino 2019   Italy Bardolino            Cavalchina
## 5     Ried Scheibner Pinot Noir 2016 Austria Carnuntum           Markowitsch
## 6   Gigondas (Nobles Terrasses) 2017  France  Gigondas         Vieux Clocher
##   Rating NumberOfRatings Price Year
## 1    4.2             100 95.00 2011
## 2    4.3             100 15.50 2017
## 3    3.9             100  7.45 2015
## 4    3.5             100  8.72 2019
## 5    3.9             100 29.15 2016
## 6    3.7             100 19.90 2017
```

```
tail(wine)
```

```
##                           Name     Country      Region             Winery Rating
## 13829   Blanco (Verdejo) 2018       Spain       Rueda Marqués de Riscal    3.7
## 13830    Sauvignon Blanc 2019 New Zealand Marlborough         Oyster Bay    4.0
## 13831  Vinho Verde Sweet N.V.    Portugal Vinho Verde       Casal Garcia    4.0
## 13832    Sauvignon Blanc 2018 New Zealand Marlborough       Kim Crawford    3.9
## 13833    Sauvignon Blanc 2019 New Zealand Marlborough        Hans Greyl    4.2
## 13834 Vinho Verde Branco N.V.    Portugal Vinho Verde       Casal Garcia    3.7
##       NumberOfRatings Price Year
## 13829            4155  6.30 2018
## 13830            4423 10.66 2019
## 13831            4609  5.05 N.V.
## 13832            5105 14.90 2018
## 13833            5817  7.75 2019
## 13834           62980  4.35 N.V.
```

```
str(wine)
```

```
## 'data.frame':    13834 obs. of  8 variables:
##  $ Name           : chr  "Pomerol 2011" "Lirac 2017" "Erta e China Rosso di Toscana 2015" "Bardolino
##  $ Country        : chr  "France" "France" "Italy" "Italy" ...
##  $ Region         : chr  "Pomerol" "Lirac" "Toscana" "Bardolino" ...
##  $ Winery         : chr  "Château La Providence" "Château Mont-Redon" "Renzo Masi" "Cavalchina" ...
```

```
##  $ Rating         : num  4.2 4.3 3.9 3.5 3.9 3.7 4 3.9 3.6 3.5 ...
##  $ NumberOfRatings: int  100 100 100 100 100 100 100 100 100 100 ...
##  $ Price          : num  95 15.5 7.45 8.72 29.15 ...
##  $ Year           : chr  "2011" "2017" "2015" "2019" ...
```

Just for fun, let's find some max and min data from the super-dataset.

```
print(max(wine$Rating))
```

```
## [1] 4.9
```

```
print(min(wine$Rating))
```

```
## [1] 2.2
```

```
print(max(wine$Price))
```

```
## [1] 3410.79
```

```
print(min(wine$Price))
```

```
## [1] 3.15
```

We have some pretty nice wine here… and some pretty awful selections as well. Though these values don't tell us much. Let's take a closer look at some of these values.

```
print(subset(wine, Rating == max(Rating)))
```

```
##                                                  Name Country
## 10612 Montrachet Grand Cru Marquis de Laguiche 2017  France
##                      Region         Winery Rating NumberOfRatings  Price Year
## 10612 Montrachet Grand Cru Joseph Drouhin    4.9              34 681.37 2017
```

```
print(subset(wine, Rating == min(Rating)))
```

```
##                            Name Country       Region   Winery Rating
## 11948 Greca Terra Retsina N.V.  Greece Peloponnesos Tsantali    2.2
##       NumberOfRatings Price Year
## 11948              77  5.35 N.V.
```

```
print(subset(wine, Price == max(Price)))
```

```
##              Name Country  Region Winery Rating NumberOfRatings   Price Year
## 2345 Pomerol 2012  France Pomerol Pétrus    4.7             204 3410.79 2012
```

```
print(subset(wine, Price == min(Price)))
```

```
##                                   Name Country Region    Winery Rating
## 9164           Frizzantino Dolce N.V.   Italy Emilia Gualtieri    4.2
## 9325 Lambrusco dell'Emilia Dolce N.V.   Italy Emilia Gualtieri    3.8
##      NumberOfRatings Price Year
## 9164              43  3.15 N.V.
## 9325             106  3.15 N.V.
```

It looks like this trivial data exploration had a purpose after all! Thanks to the Gualtieri (which looks to have two of the ultimate value choices of wine on the list) winery we find some null values in our year column. We'll need to find a way to deal with them to get a cleaner experiment done. For now, let's check how many of these values we have. We'll use the table function to see how many occurrences each value in the year column has.

```
nv_occur <- data.frame(table(wine$Year))
print(nv_occur)
```

```
##    Var1 Freq
## 1  1961    3
## 2  1988    1
## 3  1989    2
## 4  1990    2
## 5  1991    1
## 6  1992    3
## 7  1993    2
## 8  1995    4
## 9  1996    6
## 10 1997    7
## 11 1998    7
## 12 1999   19
## 13 2000   19
## 14 2001   12
## 15 2002   11
## 16 2003   15
## 17 2004   34
## 18 2005  160
## 19 2006   64
## 20 2007   60
## 21 2008  101
## 22 2009   99
## 23 2010  192
## 24 2011  312
## 25 2012  423
## 26 2013  624
## 27 2014  905
## 28 2015 1678
## 29 2016 2294
## 30 2017 2412
## 31 2018 2723
## 32 2019  893
## 33 2020    2
## 34 N.V.  744
```

Our year column seems to house a lot of "N.V." values, 744 of them. This is concerning! Where are all these N.V.'s coming from!

Recall how the totalWine.csv file was constructed. We concatenated multiple types of wine (Red, Rose, Sparkling and White) that were all in separate .csv files.

```
spark <- read.csv("Sparkling.csv", header = TRUE)
nv_occur_spark <- data.frame(table(spark$Year))
print(nv_occur_spark)
```

```
##    Var1 Freq
## 1  1961    3
## 2  1996    1
## 3  1999    2
## 4  2002    4
## 5  2003    1
## 6  2004    5
## 7  2005    2
```

```
## 8  2006   15
## 9  2007   14
## 10 2008   18
## 11 2009   13
## 12 2010   13
## 13 2011   10
## 14 2012   25
## 15 2013   16
## 16 2014   19
## 17 2015   29
## 18 2016   22
## 19 2017   26
## 20 2018   28
## 21 2019   13
## 22 N.V.  728
```

As you can see, we have a bit of an issue with our "Sparkling" section of our wine. 728/744 null values come from that section. I cannot claim to be a domain expert on wine, but a quick google search tells me that wines made from a blend of other wines will exclude a vintage date, which can explain why a lot of these sparkling wines have no dates attached to them.

This concludes an already lengthy data exploration section.

We'll copy the same conventions used in the classification section of the assignment, as we seek to make comparisons with accuracy after we complete dimensionality reduction.

Let's divide into train and test.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(dplyr)
library(ROCR)
library(mccr)
library(ISLR)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(tree)
library(rpart)
```

```
red <- read.csv("Red.csv")
white <- read.csv("White.csv")
```

```
rose <- read.csv("Rose.csv")
sparkle <- read.csv("Sparkling.csv")
totalWine <- rbind(data = red, data = white, data = rose, data = sparkle)
names(totalWine)[1] <- "Name"
```

Now we need to clean our data.

```
totalWine <- subset(totalWine, select = -c(Name, Winery, Region))
totalWine <- subset(totalWine, totalWine$Year != "N.V.")
totalWine <- subset(totalWine, totalWine$Rating != 3)
totalWine <- subset(totalWine, totalWine$Year >= 2000)
totalWine$Rating[totalWine$Rating <= 3] <- 0
totalWine$Rating[totalWine$Rating > 3] <- 1
totalWine <- totalWine[,c(1,2,3,5,4)]
```

And finally our train/test split.

```
set.seed(512)
i <- sample(1 : nrow(totalWine), round(nrow(totalWine) * 0.8), replace = FALSE)
train <- totalWine[i,]
test <- totalWine[-i,]
str(train)
```

```
## 'data.frame':    10401 obs. of  5 variables:
##  $ Country        : chr  "Austria" "Italy" "Italy" "France" ...
##  $ Rating         : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ NumberOfRatings: num  30 28 375 40 290 42 791 110 193 282 ...
##  $ Year           : chr  "2017" "2018" "2018" "2017" ...
##  $ Price          : num  13.2 12.2 14.4 9.5 69.4 ...
```

PCA

Now that our data is properly split, we'll go ahead and perform PCA on our data sets. PCA is an unsupervised dimensionality reduction algorithm, as it pays no attention to class. It reduces the data and plots them on a new coordinate space while reducing the axes. These reduced axes are principal components.

```
pca_out <- preProcess(train[,1:5],method=c("center","scale","pca"))
pca_out
```

```
## Created from 10401 samples and 5 variables
##
## Pre-processing:
##   - centered (3)
##   - ignored (2)
##   - principal component signal extraction (3)
##   - scaled (3)
##
## PCA needed 3 components to capture 95 percent of the variance
```

```
summary(pca_out)
```

```
##                   Length Class  Mode
## dim               2      -none- numeric
## bc                0      -none- NULL
## yj                0      -none- NULL
## et                0      -none- NULL
```

```
## invHyperbolicSine 0      -none- NULL
## mean              3      -none- numeric
## std               3      -none- numeric
## ranges            0      -none- NULL
## rotation          9      -none- numeric
## method            4      -none- list
## thresh            1      -none- numeric
## pcaComp           0      -none- NULL
## numComp           1      -none- numeric
## ica               0      -none- NULL
## wildcards         2      -none- list
## k                 1      -none- numeric
## knnSummary        1      -none- function
## bagImp            0      -none- NULL
## median            0      -none- NULL
## data              0      -none- NULL
## rangeBounds       2      -none- numeric
```

```
prWine <- prcomp(~Rating + NumberOfRatings + Price, data = totalWine, scale = TRUE, center = TRUE)
prWine
```

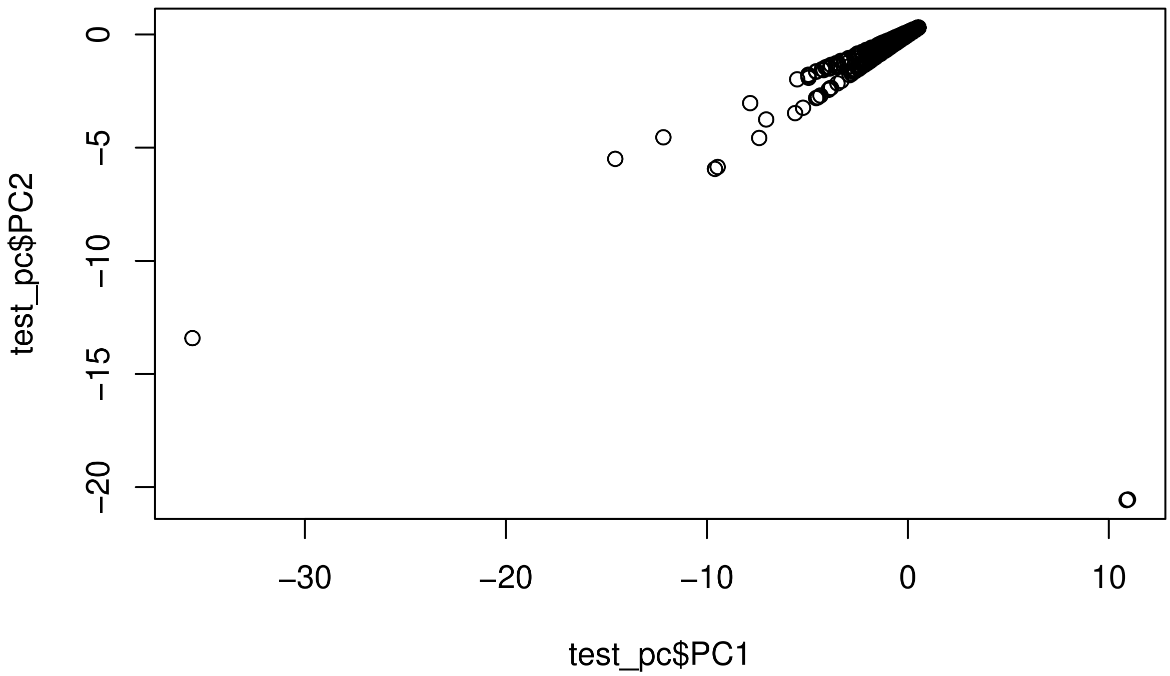
```
## Standard deviations (1, .., p=3):
## [1] 1.0172517 0.9945634 0.9879488
##
## Rotation (n x k) = (3 x 3):
##                        PC1        PC2         PC3
## Rating          -0.4884176  0.8706445 -0.05853558
## NumberOfRatings -0.6114703 -0.3893384 -0.68885393
## Price           -0.6225370 -0.3006556  0.72253294
```

As we can see, PCA found three principal components that are responsible for 95% of the variance. I will elect to leave out PC3, as a vast majority of the variance is found in PC1 and PC2. (As the first PC captures the most variance, and subsequent PCs will represent reducing variances.) Additionally, you may have noticed that we only have one attribute that is continuous (Price), this is problematic as PCA would assume that it performs it's methods on continuous variables when we've included two other discrete measures in Year and Rating. This may cause some trouble with our predictions.

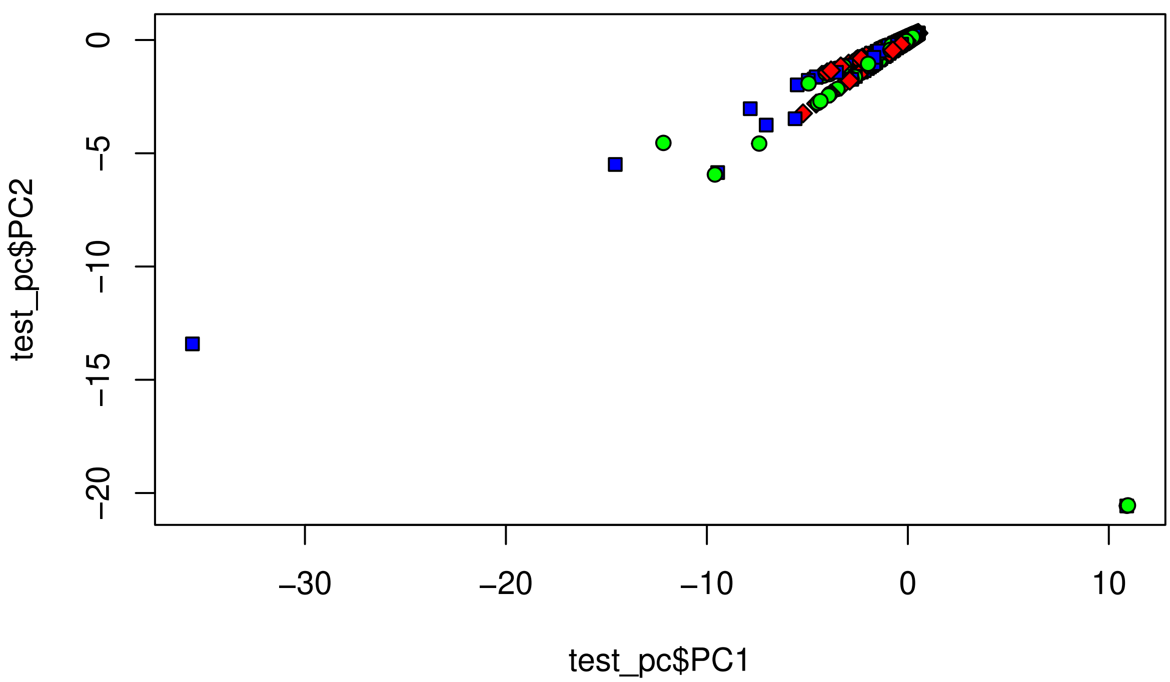
```
train_pc <- predict(pca_out, train[, 1:5])
summary(train_pc)
```

```
##    Country              Year                 PC1                 PC2
##  Length:10401       Length:10401       Min.   :-16.54772   Min.   :-20.7843
##  Class :character   Class :character   1st Qu.: -0.06901   1st Qu.:  0.0163
##  Mode  :character   Mode  :character   Median :  0.24843   Median :  0.1735
##                                        Mean   :  0.00000   Mean   :  0.0000
##                                        3rd Qu.:  0.39359   3rd Qu.:  0.2441
##                                        Max.   : 10.95378   Max.   :  0.3122
##       PC3
##  Min.   :-18.48415
##  1st Qu.: -0.15550
##  Median : -0.01497
##  Mean   :  0.00000
##  3rd Qu.:  0.13977
##  Max.   : 18.91313
```

```r
test_pc <- predict(pca_out, test[,])
plot(test_pc$PC1, test_pc$PC2)
```




```r
plot(test_pc$PC1, test_pc$PC2, pch=c(23,21,22), bg=c("red","green","blue"))
```



Now that we have done our PCA, let's try linear regression.

```r
lm1 <- lm(train_pc$PC1 ~ ., data = train_pc)
summary(lm1)
```

```
##
## Call:
```

```
## lm(formula = train_pc$PC1 ~ ., data = train_pc)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.8077  -0.0996   0.1177   0.3170  10.0909
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           -1.947571   0.247710  -7.862 4.15e-15 ***
## CountryAustralia       0.151215   0.086921   1.740 0.081944 .
## CountryAustria         0.549253   0.079132   6.941 4.13e-12 ***
## CountryBrazil          1.093705   0.173429   6.306 2.97e-10 ***
## CountryBulgaria        0.496977   0.677377   0.734 0.463161
## CountryCanada          0.707665   0.955950   0.740 0.459151
## CountryChile           0.222555   0.079948   2.784 0.005383 **
## CountryChina           3.437206   0.557904   6.161 7.50e-10 ***
## CountryCroatia         0.596618   0.481002   1.240 0.214868
## CountryCzech Republic  0.375018   0.677401   0.554 0.579856
## CountryFrance          0.241890   0.065233   3.708 0.000210 ***
## CountryGeorgia         0.615464   0.308142   1.997 0.045814 *
## CountryGermany         0.491436   0.069275   7.094 1.39e-12 ***
## CountryGreece          0.607362   0.227487   2.670 0.007600 **
## CountryHungary         0.682163   0.254011   2.686 0.007252 **
## CountryIsrael          0.422327   0.227414   1.857 0.063327 .
## CountryItaly           0.313732   0.064231   4.884 1.05e-06 ***
## CountryLebanon        -0.519928   0.300031  -1.733 0.083141 .
## CountryLuxembourg      0.624488   0.480926   1.299 0.194140
## CountryMexico          0.232642   0.956626   0.243 0.807863
## CountryMoldova        -0.062071   0.307907  -0.202 0.840241
## CountryNew Zealand     0.225620   0.104683   2.155 0.031164 *
## CountryPortugal        0.413266   0.086162   4.796 1.64e-06 ***
## CountryRomania         0.675437   0.184928   3.652 0.000261 ***
## CountrySlovakia        0.639328   0.677386   0.944 0.345286
## CountrySlovenia        0.631912   0.282277   2.239 0.025201 *
## CountrySouth Africa    0.468376   0.071791   6.524 7.16e-11 ***
## CountrySpain           0.327171   0.067503   4.847 1.27e-06 ***
## CountrySwitzerland     0.356522   0.227498   1.567 0.117111
## CountryTurkey          0.568930   0.324048   1.756 0.079170 .
## CountryUnited Kingdom  0.245637   0.554625   0.443 0.657856
## CountryUnited States  -0.167760   0.077213  -2.173 0.029826 *
## CountryUruguay        -0.532162   0.554220  -0.960 0.336978
## Year2001               1.170203   0.398062   2.940 0.003292 **
## Year2002              -0.265421   0.432899  -0.613 0.539807
## Year2003               0.553260   0.364327   1.519 0.128899
## Year2004               0.618371   0.299050   2.068 0.038685 *
## Year2005               0.664562   0.254822   2.608 0.009122 **
## Year2006               0.192919   0.276723   0.697 0.485719
## Year2007               0.884881   0.276914   3.196 0.001400 **
## Year2008               0.612363   0.262004   2.337 0.019446 *
## Year2009               0.644891   0.263526   2.447 0.014415 *
## Year2010               0.664953   0.251239   2.647 0.008141 **
## Year2011               1.276985   0.246814   5.174 2.34e-07 ***
## Year2012               1.321315   0.245142   5.390 7.20e-08 ***
## Year2013               1.383877   0.243527   5.683 1.36e-08 ***
```

```
## Year2014              1.505250   0.242455   6.208 5.56e-10 ***
## Year2015              1.542855   0.241325   6.393 1.69e-10 ***
## Year2016              1.650301   0.240959   6.849 7.87e-12 ***
## Year2017              1.782088   0.241103   7.391 1.57e-13 ***
## Year2018              1.936895   0.241028   8.036 1.03e-15 ***
## Year2019              1.994149   0.242817   8.213 2.42e-16 ***
## Year2020              1.966979   0.716857   2.744 0.006082 **
## PC2                  -0.044483   0.009606  -4.631 3.68e-06 ***
## PC3                   0.064372   0.009998   6.439 1.26e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9537 on 10346 degrees of freedom
## Multiple R-squared:  0.1316, Adjusted R-squared:  0.127
## F-statistic: 29.03 on 54 and 10346 DF,  p-value: < 2.2e-16
```

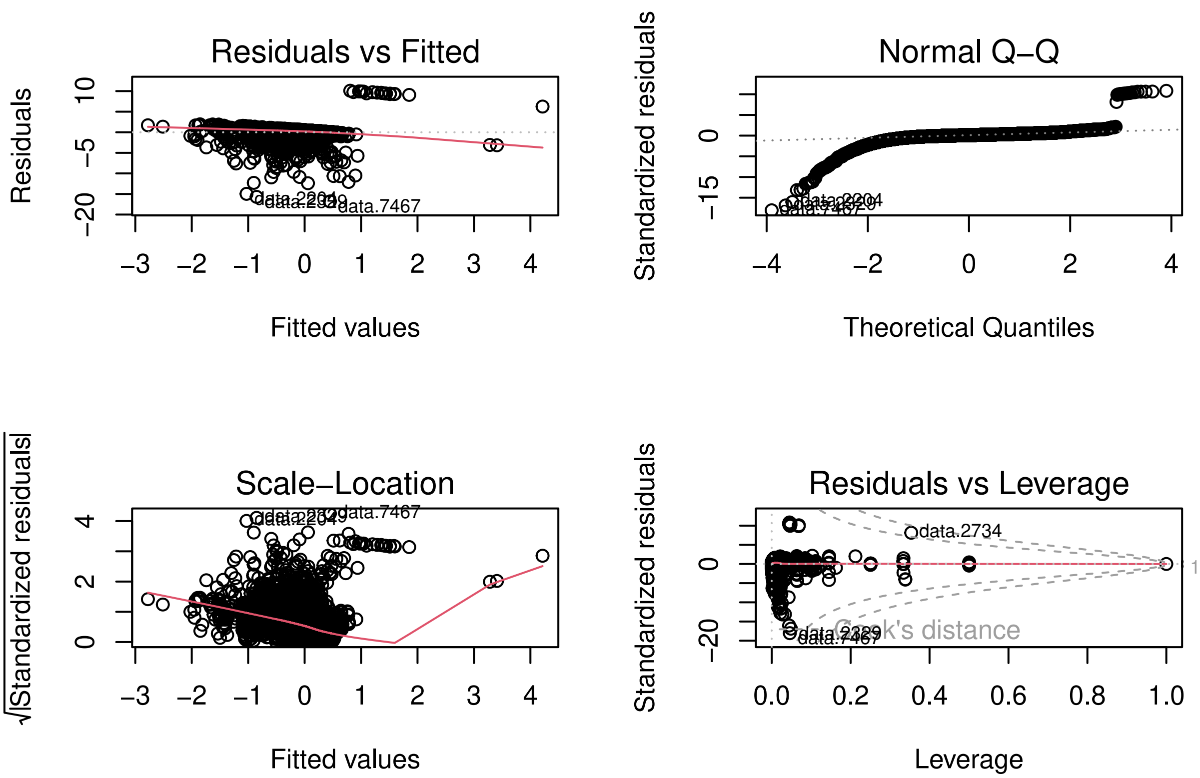
Now let's plot the residuals.

```
par(mfrow = c(2,2))
plot(lm1)
```

```
## Warning: not plotting observations with leverage one:
##   2844
```

```
## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced
```

```
## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced
```




```
pred1 <- predict(lm1, newdata = test_pc)
summary(pred1)
```

```
##      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
```

```
## -1.938770 -0.156253  0.038845 -0.006681  0.262473  2.811055
```

```r
print(cor(pred1, test_pc$PC1))
```

```
## [1] 0.1674418
```

```r
print(mean(pred1 - test_pc$PC1) ^ 2)
```

```
## [1] 7.671703e-06
```

```r
rmse1 <- sqrt(mean(pred1 - test_pc$PC1) ^ 2)
rmse1
```

```
## [1] 0.002769784
```

We can see that our correlation values fell a bit, signaling a slight reduction in accuracy. Which makes sense as we had reduced our data.

LDA

Linear discriminant analysis WILL consider class, unlike its cousin PCA. Thus, LDA can be superior when we have a known class to fit data into. LDA finds a linear combination of predictors that maximizes separation between classes and minimizes the standard deviation within a class.

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```r
lda1 <- lda(Price~., data = train)
lda1$means
```

```
##          CountryAustralia CountryAustria CountryBrazil CountryBulgaria
## 3.55           0.00000000     0.00000000    0.00000000       0.0000000
## 3.7            0.00000000     0.00000000    0.00000000       0.0000000
## 3.74           0.00000000     0.00000000    0.00000000       0.0000000
## 3.75           0.00000000     0.00000000    0.00000000       0.0000000
## 3.79           0.00000000     0.00000000    0.00000000       0.0000000
## 3.95           0.00000000     0.00000000    0.00000000       0.0000000
## 3.99           0.00000000     0.00000000    0.00000000       0.0000000
## 4              0.00000000     0.00000000    0.00000000       0.0000000
## 4.15           0.00000000     0.00000000    0.00000000       0.0000000
## 4.16           0.00000000     0.00000000    0.00000000       0.0000000
## 4.25           0.00000000     0.00000000    0.00000000       0.0000000
## 4.28           0.00000000     0.00000000    0.00000000       0.0000000
## 4.29           0.00000000     0.00000000    0.00000000       0.0000000
## 4.3            0.00000000     0.00000000    0.00000000       0.0000000
## 4.31           0.00000000     0.00000000    0.00000000       0.0000000
## 4.38           0.00000000     0.00000000    0.00000000       0.0000000
## 4.4            0.00000000     0.00000000    0.00000000       0.0000000
## 4.44           0.00000000     0.00000000    0.00000000       0.0000000
## 4.45           0.00000000     0.00000000    0.00000000       0.0000000
## 4.5            0.00000000     0.00000000    0.00000000       0.0000000
## 4.55           0.00000000     0.00000000    0.00000000       0.0000000
```

```
## 4.57          0.00000000     0.00000000    0.00000000       0.0000000
## 4.6           0.00000000     0.00000000    0.00000000       0.0000000
## 4.63          0.00000000     0.00000000    0.00000000       0.0000000
## 4.65          0.00000000     0.00000000    0.00000000       0.0000000
## 4.66          0.00000000     0.00000000    0.00000000       0.0000000
## 4.67          0.00000000     0.00000000    0.00000000       0.0000000
## 4.7           0.00000000     0.00000000    0.00000000       0.0000000
## 4.73          0.00000000     0.00000000    0.00000000       0.0000000
## 4.74          0.00000000     0.00000000    0.00000000       0.0000000
## 4.75          0.00000000     0.00000000    0.00000000       0.0000000
## 4.76          0.00000000     0.00000000    0.00000000       0.0000000
## 4.78          0.00000000     0.00000000    0.00000000       0.0000000
## 4.79          0.00000000     0.00000000    0.00000000       0.0000000
## 4.8           0.00000000     0.00000000    0.00000000       0.0000000
## 4.82          0.00000000     0.00000000    0.00000000       0.0000000
## 4.83          0.00000000     0.00000000    0.00000000       0.0000000
## 4.84          0.00000000     0.00000000    0.00000000       0.0000000
## 4.85          0.00000000     0.00000000    0.00000000       0.0000000
## 4.86          0.00000000     0.00000000    0.00000000       0.0000000
## 4.89          0.00000000     0.00000000    0.00000000       0.0000000
## 4.9           0.00000000     0.00000000    0.00000000       0.0000000
## 4.92          0.00000000     0.00000000    0.00000000       0.0000000
## 4.94          0.00000000     0.00000000    0.00000000       0.0000000
## 4.95          0.00000000     0.00000000    0.00000000       0.0000000
## 4.99          0.00000000     0.00000000    0.00000000       0.0000000
## 5             0.00000000     0.00000000    0.00000000       0.0000000
## 5.02          0.00000000     0.00000000    0.00000000       0.0000000
## 5.03          0.00000000     0.00000000    0.00000000       0.0000000
## 5.1           0.00000000     0.00000000    0.00000000       0.0000000
## 5.12          0.00000000     0.00000000    0.00000000       0.0000000
## 5.14          0.00000000     0.00000000    0.00000000       0.0000000
## 5.15          0.00000000     0.00000000    0.00000000       0.0000000
## 5.16          0.00000000     0.00000000    0.00000000       0.0000000
## 5.2           0.00000000     0.00000000    0.00000000       0.0000000
## 5.21          0.00000000     0.00000000    0.00000000       0.0000000
## 5.23          0.00000000     0.00000000    0.00000000       0.0000000
## 5.25          0.00000000     0.00000000    0.00000000       0.0000000
## 5.28          0.00000000     0.00000000    0.00000000       0.0000000
## 5.29          0.00000000     0.00000000    0.00000000       0.0000000
## 5.3           0.00000000     0.00000000    0.00000000       0.0000000
## 5.31          0.00000000     0.00000000    0.00000000       0.0000000
## 5.32          0.00000000     0.00000000    0.00000000       0.0000000
## 5.33          0.00000000     0.00000000    0.00000000       0.0000000
## 5.35          0.00000000     0.00000000    0.00000000       0.0000000
## 5.36          0.00000000     0.00000000    0.00000000       0.0000000
## 5.38          0.00000000     0.00000000    0.00000000       0.0000000
## 5.4           0.00000000     0.00000000    0.00000000       0.0000000
## 5.42          0.00000000     0.00000000    0.00000000       0.0000000
## 5.43          0.00000000     0.00000000    0.00000000       0.0000000
## 5.45          0.00000000     0.00000000    0.00000000       0.0000000
## 5.46          0.00000000     0.00000000    0.00000000       0.0000000
## 5.49          0.00000000     0.00000000    0.00000000       0.0000000
## 5.5           0.00000000     0.00000000    0.00000000       0.0000000
## 5.51          0.00000000     0.00000000    0.00000000       0.0000000
```

```
## 5.52           0.00000000     0.00000000    0.00000000       0.0000000
## 5.53           0.00000000     0.00000000    0.00000000       0.0000000
## 5.54           0.00000000     0.00000000    0.00000000       0.0000000
## 5.55           0.00000000     0.00000000    0.00000000       0.0000000
## 5.56           0.00000000     0.00000000    0.00000000       0.0000000
## 5.59           0.00000000     0.00000000    0.00000000       0.0000000
## 5.6            0.00000000     0.00000000    0.00000000       0.0000000
## 5.61           0.00000000     0.00000000    0.00000000       0.0000000
## 5.62           0.00000000     0.00000000    0.00000000       0.0000000
## 5.63           0.00000000     0.00000000    0.00000000       0.0000000
## 5.64           0.00000000     0.00000000    0.00000000       0.0000000
## 5.65           0.00000000     0.16666667    0.00000000       0.0000000
## 5.68           0.00000000     0.00000000    0.00000000       0.0000000
## 5.69           0.00000000     0.00000000    0.00000000       0.0000000
## 5.7            0.00000000     0.00000000    0.00000000       0.0000000
## 5.71           0.00000000     0.00000000    0.00000000       0.0000000
## 5.72           0.00000000     0.00000000    0.00000000       0.0000000
## 5.74           0.00000000     0.00000000    0.00000000       0.0000000
## 5.75           0.00000000     0.00000000    0.00000000       0.0000000
## 5.76           0.00000000     0.00000000    0.00000000       0.0000000
## 5.77           0.00000000     0.00000000    0.00000000       0.0000000
## 5.79           0.16666667     0.00000000    0.00000000       0.0000000
## 5.8            0.00000000     0.00000000    0.00000000       0.0000000
## 5.81           0.00000000     0.00000000    0.00000000       0.0000000
## 5.82           0.00000000     0.00000000    0.00000000       0.0000000
## 5.84           0.00000000     0.00000000    0.00000000       0.0000000
## 5.85           0.00000000     0.00000000    0.00000000       0.0000000
## 5.89           0.00000000     0.00000000    0.00000000       0.0000000
## 5.9            0.00000000     0.00000000    0.00000000       0.0000000
## 5.94           0.00000000     0.00000000    0.00000000       0.0000000
## 5.95           0.06060606     0.00000000    0.00000000       0.0000000
## 5.98           0.00000000     0.00000000    0.00000000       0.0000000
## 5.99           0.00000000     0.00000000    0.00000000       0.0000000
## 6              0.00000000     0.00000000    0.00000000       0.0000000
## 6.02           0.00000000     0.00000000    0.00000000       0.0000000
## 6.03           0.00000000     0.00000000    0.00000000       0.0000000
## 6.05           0.00000000     0.00000000    0.00000000       0.0000000
## 6.09           0.00000000     0.00000000    0.00000000       0.0000000
## 6.1            0.00000000     0.00000000    0.00000000       0.0000000
## 6.12           0.00000000     0.00000000    0.00000000       0.0000000
## 6.13           0.00000000     0.00000000    0.00000000       0.0000000
## 6.15           0.00000000     0.00000000    0.00000000       0.0000000
## 6.16           0.00000000     0.00000000    0.00000000       0.0000000
## 6.19           0.00000000     0.00000000    0.00000000       0.0000000
## 6.2            0.00000000     0.00000000    0.00000000       0.0000000
## 6.21           0.25000000     0.00000000    0.00000000       0.0000000
## 6.23           0.00000000     0.00000000    0.00000000       0.0000000
## 6.24           0.00000000     0.00000000    0.00000000       0.0000000
## 6.25           0.00000000     0.00000000    0.00000000       0.0000000
## 6.26           0.00000000     0.00000000    0.00000000       0.0000000
## 6.29           0.00000000     0.00000000    0.00000000       0.0000000
## 6.3            0.00000000     0.00000000    0.00000000       0.0000000
## 6.31           0.00000000     0.00000000    0.00000000       0.0000000
## 6.32           0.00000000     0.00000000    0.00000000       0.0000000
```

```
## CountryLebanon        -6.787403e-02  4.172415e-02 -1.868567e-02  3.909762e-02
## CountryLuxembourg     -2.704892e-02  8.856544e-02  5.681759e-03 -5.250638e-02
## CountryMexico         -4.400869e-02  2.745469e-01  4.512722e-02 -1.461833e-01
## CountryMoldova        -8.066841e-04  3.709996e-02 -2.759620e-03 -4.596180e-02
## CountryNew Zealand     1.511986e-01 -1.240325e-01  5.884640e-02 -1.357162e-01
## CountryPortugal       -3.593999e-01  9.975121e-02 -2.041191e-02 -9.821181e-03
## CountryRomania         5.381921e-01  1.496702e+01 -2.093963e+00 -5.114234e+00
## CountrySlovakia        1.017742e+01  2.115199e+01  1.649101e+01  5.629167e+01
## CountrySlovenia       -1.545114e+01  5.468643e-01  1.279180e-01  1.307837e+00
## CountrySouth Africa   -1.687347e-02 -1.377714e-01  6.983389e-02 -1.157027e-01
## CountrySpain           1.381214e-02  4.197080e-02  1.440170e-02 -1.981445e-02
## CountrySwitzerland    -4.758196e-02 -6.645005e-02  1.635612e-02  2.324576e-02
## CountryTurkey         -3.918113e+00 -5.327550e+00  6.140311e-01  2.011485e+00
## CountryUnited Kingdom -1.613954e-01  1.692158e-02 -2.639647e-02  4.376702e-02
## CountryUnited States   1.009736e-02 -1.270576e-02  3.299893e-02 -2.269264e-02
## CountryUruguay         2.058015e+00  5.830943e-01 -1.785684e-02  1.687805e-01
## Rating                -1.585820e-01  2.854437e-01  2.956890e-02 -1.394051e-01
## NumberOfRatings        6.344999e-06 -7.961667e-06  6.032259e-06 -3.271830e-06
## Year2001              -3.816442e-02  8.939072e-03  2.504713e-03  1.387763e-02
## Year2002              -1.446647e-02  1.310904e-03 -6.480272e-03  6.999415e-03
## Year2003               4.525561e-02  1.073489e-01 -3.577545e-03 -3.193922e-02
## Year2004              -1.790243e-02 -5.212768e-03  1.047114e-02 -2.709072e-03
## Year2005              -2.473786e-02  9.863940e-03 -1.232727e-02  1.420073e-02
## Year2006              -5.968087e-03  1.955486e-03  2.529429e-03  3.054409e-03
## Year2007               1.557529e-03  1.905899e-02 -5.753306e-03  4.333951e-03
## Year2008               3.543613e-02  1.442041e-02 -2.189123e-02  5.185512e-03
## Year2009               5.612026e-02 -8.708113e-03 -1.508398e-02  1.529873e-02
## Year2010              -2.867112e-02  1.458319e-02 -1.264813e-02 -1.797924e-02
## Year2011               3.562875e-02  1.598819e-02  4.142401e-03 -4.502524e-02
## Year2012               3.155637e-02  3.473633e-02 -2.707898e-03 -9.261088e-03
## Year2013               6.469553e-02  2.945535e-03 -7.661883e-03 -4.013230e-02
## Year2014               5.722172e-02  2.471491e-02  7.339117e-03 -2.829255e-02
## Year2015              -2.736074e-02  2.449592e-02  4.385395e-02 -4.777540e-02
## Year2016               2.099765e-03 -2.377244e-02 -2.187815e-03 -5.874881e-03
## Year2017               3.162700e-02 -8.278620e-02  1.043988e-02  1.028478e-02
## Year2018              -5.897151e-03 -6.109832e-02  2.134442e-02  3.431160e-02
## Year2019               4.314965e-02 -4.247977e-02 -1.300157e-02  1.096318e-02
## Year2020              -1.827492e-02  2.319103e+00 -1.000807e+00 -1.224477e+00
##                                LD53          LD54
## CountryAustralia      -2.639318e-02 -8.469747e-04
## CountryAustria        -4.254561e-02  4.696618e-02
## CountryBrazil          8.474102e-03  1.140675e-01
## CountryBulgaria       -1.779625e-02  3.236773e-03
## CountryCanada         -6.280130e-02 -5.827562e-01
## CountryChile          -1.806197e-03  4.084109e-02
## CountryChina          -2.706210e-03  5.098557e-02
## CountryCroatia         3.452416e+01  4.640353e+00
## CountryCzech Republic -3.816555e+01 -5.049928e+00
## CountryFrance         -4.629070e-02  1.450437e-02
## CountryGeorgia        -2.627039e-03  5.555491e-02
## CountryGermany        -3.170398e-02  1.660121e-02
## CountryGreece          1.222406e+00  3.616305e-01
## CountryHungary        -9.920278e-01  3.979240e-02
## CountryIsrael          2.337067e-02  9.713556e-02
```

```
## CountryItaly          -2.011163e-02  8.057530e-03
## CountryLebanon        -1.644750e-02 -2.232072e-03
## CountryLuxembourg      3.403336e-03  2.963174e-02
## CountryMexico          3.008343e-03  1.771995e-01
## CountryMoldova        -2.229544e-02  2.285581e-02
## CountryNew Zealand    -5.762401e-02  6.604469e-02
## CountryPortugal       -6.333612e-03  1.715160e-02
## CountryRomania        -9.793819e-01 -7.624314e-01
## CountrySlovakia        3.089524e+00  1.158605e+00
## CountrySlovenia        3.951880e-01  2.300480e-01
## CountrySouth Africa   -7.885672e-03  3.057292e-02
## CountrySpain          -1.490683e-02  3.096824e-02
## CountrySwitzerland    -4.966276e-03  5.185219e-02
## CountryTurkey          3.415580e-01  4.412937e-01
## CountryUnited Kingdom  4.329752e-02  1.543064e-02
## CountryUnited States  -2.069601e-03  1.681490e-02
## CountryUruguay        -3.384096e+00 -3.025339e-01
## Rating                -1.144184e-01  5.404661e-01
## NumberOfRatings       -5.374291e-06  8.479286e-06
## Year2001               7.256051e-03 -2.248157e-03
## Year2002               3.611548e-03 -3.296748e-03
## Year2003               4.948091e-03 -4.626497e-03
## Year2004               2.712387e-03  4.321348e-03
## Year2005               5.149756e-03 -1.523470e-03
## Year2006               2.077651e-03  9.355211e-04
## Year2007               4.126353e-03  1.113096e-02
## Year2008               4.609467e-02  3.971207e-03
## Year2009              -7.496749e-02 -6.902149e-03
## Year2010              -1.210951e-02  3.869984e-03
## Year2011              -1.613261e-02 -1.684513e-02
## Year2012               1.059868e-03  6.754828e-03
## Year2013              -2.417498e-02  6.475244e-03
## Year2014              -1.843354e-03  1.066015e-02
## Year2015              -7.477242e-03 -4.137931e-04
## Year2016              -2.124870e-02  3.678406e-03
## Year2017              -5.526997e-02  5.553615e-03
## Year2018               3.306930e-03 -1.827223e-02
## Year2019              -1.208776e-02 -1.504149e-01
## Year2020              -8.626739e+00  6.230472e+01
```

Now let's test our LDA1 model.

```r
lda_pred <- predict(lda1, newdata = test, type = "class")
options(max.print = 100) #the output here was excessive, so I had to cut it down slightly.
lda_pred$class
```

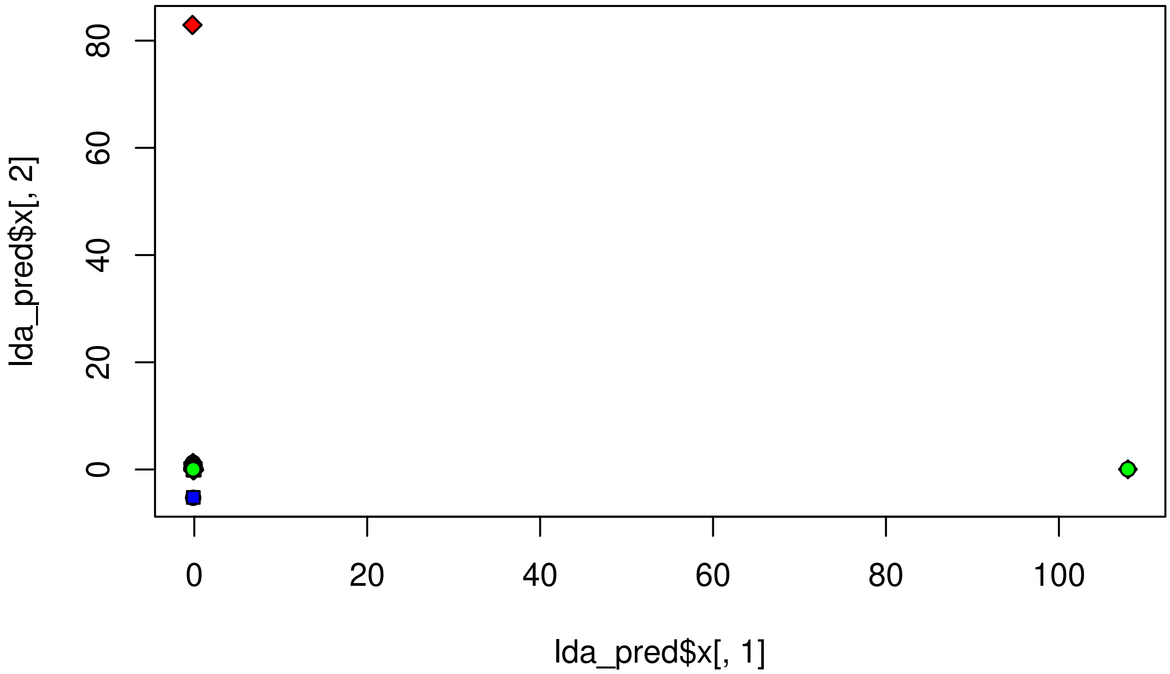
```
##   [1] 129.01 32.14  15.5   30.74  15.5   140.64 43.16  32.75  8.9    54.5
##  [11] 5.36   32.75  15.5   75.07  7.42   37.73  161.1  120.01 156.17 36.89
##  [21] 9.34   43.16  58.8   7.37   36.83  7.84   100    45.77  5.36   130.97
##  [31] 36.83  15.5   5.31   129.01 26.06  18.4   484.03 19.24  7.65   63.26
##  [41] 15.5   43.16  32.75  7.42   5.31   25.84  328.95 5.99   58.8   7.42
##  [51] 129.01 62.39  62.39  328.95 29.1   20.19  36.63  139.95 129.01 11.56
##  [61] 130.97 28.6   43.16  39.89  125.65 28.41  7.42   12.98  208.44 5.36
##  [71] 43.16  5.36   7.84   36.63  22.85  20.19  7.42   54.59  32.75  5.99
##  [81] 29.1   9.34   7.84   23.38  18.4   3.74   137.58 137.58 29.12  76.76
```

```
##  [91] 63.66  130.97 15.5   721.34 19.05  8.9    484.03 484.03 17.42  129.01
##  [ reached getOption("max.print") -- omitted 2500 entries ]
## 2616 Levels: 3.55 3.7 3.74 3.75 3.79 3.95 3.99 4 4.15 4.16 4.25 4.28 4.29 ... 1599.95
```

```r
mean(lda_pred$class == test$Price)
```

```
## [1] 0.005
```

```r
plot(lda_pred$x[,1], lda_pred$x[,2], pch=c(23,21,22), bg=c("red","green","blue"))
```



With a mean of 0.005, we have an absolutely diabolical accuracy score.