LDA and Dimension Reduction

Libraries

```
library(MASS) ## For  lda
library(tidyverse)
```

```
## -- Attaching packages --------- tidyverse 1.2.1 --

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.3.0

## -- Conflicts ------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::select() masks MASS::select()
```

Introduction

Use all you know about LDA (and QDA) to analyze a data set related to classification vehicle types.

The data are originally from the UCI Machine Learning Data. It can be loaded via the R **mlbench** library

The Vehicle data set contains 848 observations. There are 18 predictors and one response variable.
* The 18 predictors are measurements of types of cars. * The response Class is a car type: bus, van, opel, saab.

Of course, look at this data set, poke it, probe, it peek at it.
The goal is to classify the car type based on its measurements. As well, we want to see what sort of dimension reduction we can perform in order to visualize the effectiveness of the classification.

More information on the variables in the Vehicles dataset is available at the UCI Machine Learning Repository description.

There are four types of cars. Take a peek at the Class distribution.

```
with(Vehicle,table(Class))
```

```
## Class
##  bus opel saab  van
##  218  212  217  199
```

Starter: Only two car types.

Pick two car types to get things rolling, say “opel” and “saab”.

```
Vehicle2.df <- Vehicle %>%
  filter(Class %in% c("opel","saab")) %>%
  droplevels() ##ignore the other Class factors
```

Task: Use both logistic regression and LDA to predict between these two class. Since we are going to appraise the effectiveness of the model create train/test data frames (of equal size). As always, build the model on the training data, predict on the testing data.

```
N <- nrow(Vehicle2.df)
train <- sample(1:N,N/2,rep=F)
train.df <- Vehicle2.df[train,]
test.df <- Vehicle2.df[-train,]
```

Logistic Regression.

As a first comparison, model the using logistic regression (use train/test)!

Create response variable called **class.log** from the predictions in the test data frame.

Compute the confusion matrix and the error rate.

```
##       Class.log
## Class  opel saab
##   opel   67   41
##   saab   35   72
```

```
## [1] 0.3534884
```

Now model with LDA. Again compute the confusion matrix and the error rate. Create response variable called **class.lda** from the predictions in the test data frame.

Compute the confusion matrix and the error rate.

```
##       Class.lda
## Class  opel saab
##   opel   68   40
##   saab   36   71
```

```
## [1] 0.3534884
```

Looks like both models are doing about the same with respect to prediction.

How do the models compare with respect to the classifications?

```
##          Class.lda
## Class.log opel saab
##      opel  102    0
##      saab    2  111
```

Not surprisingly, pretty close.

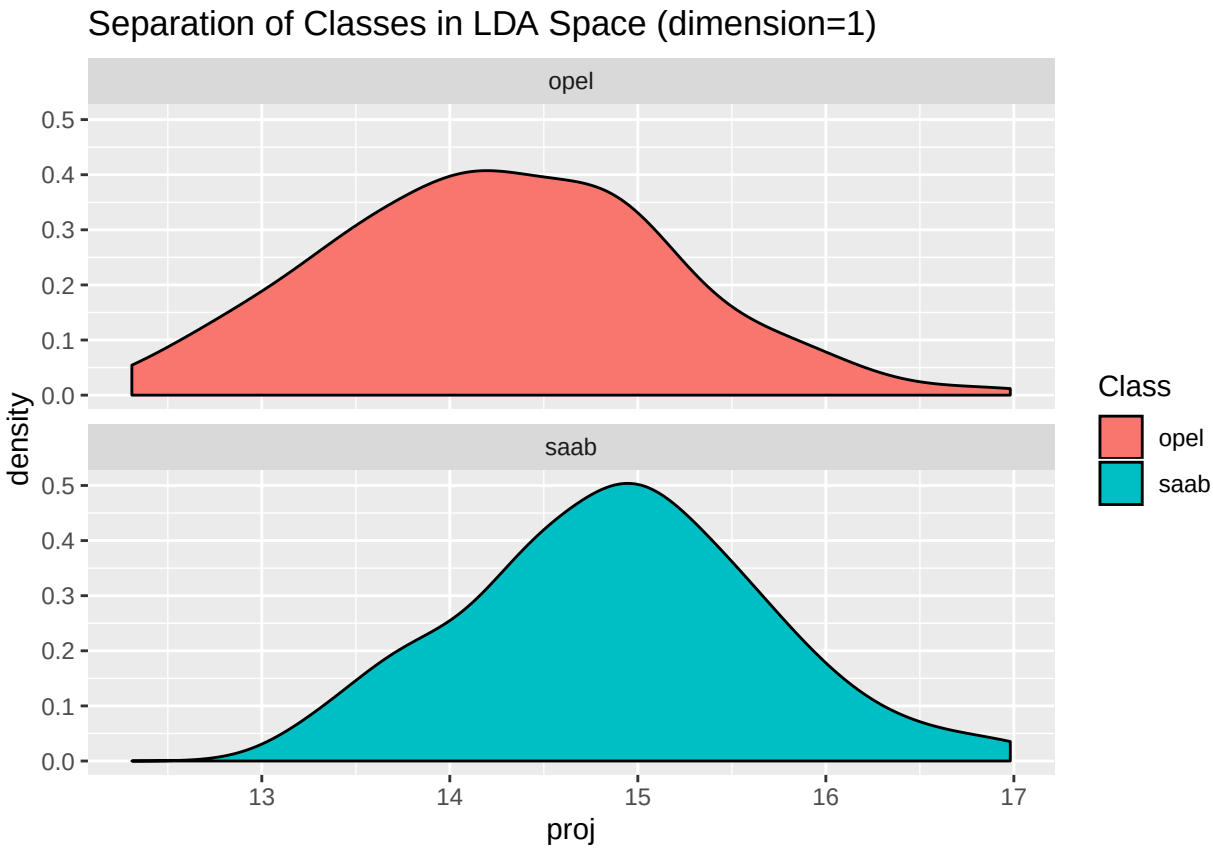
LDA and Scaling

The predictors are in R^{18}, a very high dimensional space. Use the scaling vector to project the data onto the one-dimensional projection space.

Here is what the scaling vector looks like (your vector can be slightly different owing to the train/test randomness).

```
##                     LD1
## Comp          0.08984798
## Circ         -0.30727542
## D.Circ       -0.01410910
## Rad.Ra        0.03743334
## Pr.Axis.Ra   -0.08122937
## Max.L.Ra     -0.05126562
## Scat.Ra       0.05639587
## Elong         0.03008074
## Pr.Axis.Rect  0.77989946
## Max.L.Rect    0.01344223
## Sc.Var.Maxis -0.01787718
## Sc.Var.maxis -0.02543260
## Ra.Gyr        0.03930538
## Skew.Maxis    0.05479998
## Skew.maxis    0.04253088
## Kurt.maxis   -0.04074433
## Kurt.Maxis   -0.38272601
## Holl.Ra       0.34454422
```

Grab the data matrix and project the observations onto the scaling matrix. When you have done this, plot the densities of each class to get a visual idea of how separated they are.



Other Classes

Repeat the above analysis with "bus" and "van" instead of "opel" and "saab".

Repeat the above analysis with "opel" and "bus" instead of "opel" and "saab".

Three Classes: LDA Only

Now redo the process above with C=3 classes. LDA with C=3 classes allows us to project the results onto C-1=2 dimensional space. That's what we'll do here.

Use any three of the Classes to define your 3-Class data set.

```
Vehicle3.df <- Vehicle %>%
  filter(Class %in% c("opel","saab","van")) %>%
  droplevels()
```

Note: it is quite difficult to effectively use logistic regression with C=3 classes.

Plan:

- Create train/test data frames.
- Model with LDA on the trainging data.

- Since C=3, there will be C-1=2 scaling vectors. mod.lda$scaling will be a matrix with two columns. Each column is a scaling vector. Extract these separately and normalize as before.
- Project the test prediction variables onto both of these normalized scaling vectors (proj1 and proj2).
- How does the class separation look in the (proj1,proj2) space?

Model with LDA and compute the confusion matrix and error rate

```
##       Class.lda
## Class  opel saab van
##   opel   65   43   0
##   saab   26   75   0
##   van     1    7  97
```

```
## [1] 0.2452229
```

Now extract the scaling matrix. In this case, the scaling matrix had dimension 18 x 2. After you have done so, separate the two columns and use these vectors as your scaling vectors. Of course, normalize.

As before, project the observations onto the scaling directions. Now you will have two projection directions, proj1 and proj2.

```
##                       LD1           LD2
## Comp          0.107628587 -1.874002e-01
## Circ         -0.258891727  3.808109e-01
## D.Circ        0.112860448  5.075092e-02
## Rad.Ra       -0.094873744 -1.253718e-02
## Pr.Axis.Ra    0.258535922 -1.863969e-03
## Max.L.Ra     -0.139100416  1.141373e-01
## Scat.Ra      -0.014587171 -8.365216e-02
## Elong         0.197163999 -9.212548e-02
## Pr.Axis.Rect -0.365059846 -7.811072e-01
## Max.L.Rect    0.194147821 -8.282956e-06
## Sc.Var.Maxis  0.060895965 -4.642214e-02
## Sc.Var.maxis -0.006460240  3.353490e-02
## Ra.Gyr       -0.009105751 -5.094765e-02
## Skew.Maxis    0.099570258 -3.093028e-02
## Skew.maxis   -0.020219553 -9.291590e-03
## Kurt.maxis   -0.031957352  1.783405e-02
## Kurt.Maxis   -0.034600895  3.216392e-01
## Holl.Ra       0.165412727 -2.900966e-01
```

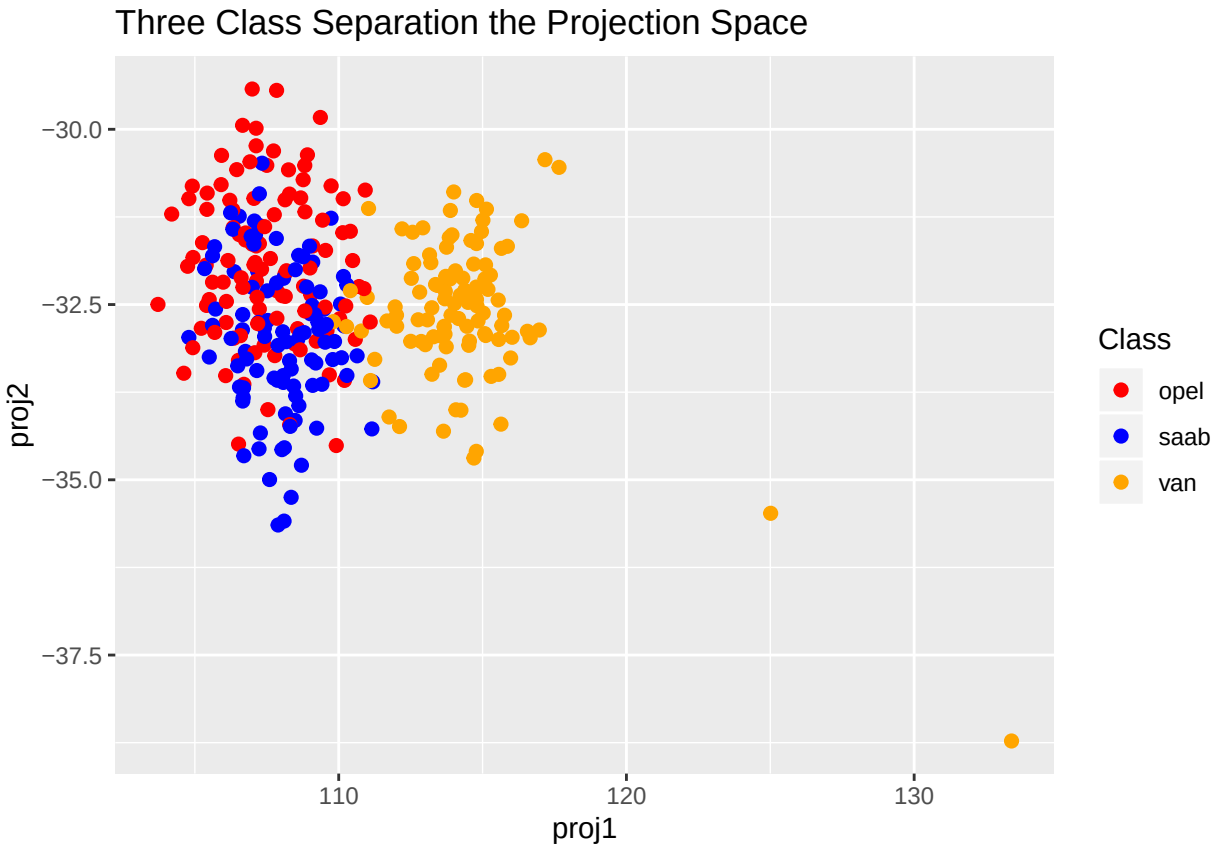
As reference, you should get something like this as your scaling vectors.

```
cbind(lda.scale1,lda.scale2)
```

```
##                lda.scale1    lda.scale2
## Comp          0.162196955 -1.858861e-01
## Circ         -0.390151456  3.777342e-01
## D.Circ        0.170081403  5.034088e-02
## Rad.Ra       -0.142975327 -1.243589e-02
## Pr.Axis.Ra    0.389615255 -1.848909e-03
```

```
## Max.L.Ra     -0.209625198  1.132151e-01
## Scat.Ra      -0.021982958 -8.297630e-02
## Elong         0.297127383 -9.138116e-02
## Pr.Axis.Rect -0.550147477 -7.747963e-01
## Max.L.Rect    0.292581983 -8.216035e-06
## Sc.Var.Maxis  0.091770601 -4.604707e-02
## Sc.Var.maxis -0.009735622  3.326396e-02
## Ra.Gyr       -0.013722423 -5.053603e-02
## Skew.Maxis    0.150053003 -3.068038e-02
## Skew.maxis   -0.030470993 -9.216520e-03
## Kurt.maxis   -0.048159930  1.768996e-02
## Kurt.Maxis   -0.052143766  3.190406e-01
## Holl.Ra       0.249278016 -2.877528e-01
```

What does this look like with in the 2-dimensional projections space? Create a plot that shows the separation in the (proj1, proj) space.



Comment on what you see here.

Repeat for another subset of the car types.

Four Classes

Now model in the full dataset using all four classes. Everything is the same as before, only there is more of it.

Compute the class predictions and error rate.

Visualization is challenging since the the projection space is 3-dimensional. Think about how you can visually present your results in a way.

```
##       Class.lda
## Class  bus opel saab van
##   bus  105    1    0   3
##   opel   4   64   35   4
##   saab  10   31   64   8
##   van    1    4    0  89
```

```
## [1] 0.2387707
```

```
##                         LD1          LD2         LD3
## Comp          -0.0259724516  0.097459071 -0.20610199
## Circ          -0.1225567526 -0.550397284  0.15644923
## D.Circ         0.0323765399  0.110174983  0.07862526
## Rad.Ra         0.1318297851 -0.042367634  0.02828737
## Pr.Axis.Ra    -0.3733334999  0.120827752 -0.10062391
## Max.L.Ra       0.1898016368 -0.135894530  0.12377515
## Scat.Ra       -0.0269995584 -0.267492478 -0.17700679
## Elong          0.3002512054  0.077452535 -0.08552047
## Pr.Axis.Rect   0.4810697811  0.045188277 -0.96381040
## Max.L.Rect     0.0023381021  0.271042359  0.06006882
## Sc.Var.Maxis  -0.0070050065  0.025299417 -0.05290220
## Sc.Var.maxis  -0.0035409274  0.033981099  0.04922841
## Ra.Gyr         0.0002762928  0.005357441 -0.04801763
## Skew.Maxis    -0.0566119356  0.073157006  0.03560232
## Skew.maxis     0.0528708723 -0.014400675  0.01915224
## Kurt.maxis    -0.0323879154 -0.024864923  0.02195427
## Kurt.Maxis    -0.3633403602 -0.036107977  0.17855355
## Holl.Ra        0.2267384134  0.072464291 -0.17327941
```

ggplot in 3-D

Note You might want to try a 3-dimensional visualization. R and ggplot are essentially 2-dimensional. However, there is a package called **gg3D** which is under development. It can be installed using the R **devtools** (you can, should, install this).

```
devtools::install_github("AckerDWM/gg3D")
```

```
## Skipping install of 'gg3D' from a github remote, the SHA1 (ffdd837d) has not changed since last insta
##   Use `force = TRUE` to force installation
```

```
library("gg3D")
```

Info on how to use gg3D is available online.

```
cols <- c("red","blue","orange")
test.df %>%
  ggplot( aes(x=proj1, y=proj2, z=proj3, color=factor(Class))) +
  theme_void() +
```

```
  axes_3D(theta=110,phi=10) +
scale_color_manual(values=c(cols,"cyan"))+
stat_3D()+
labs(title="3 Dimensional Projection Space",
     color="Class")
```