

Classification

L. Insolia, F. Chiaromonte (special thanks to J. Di Iorio)

March 22nd 2022

Contents

Introduction	1
Libraries	1
Data	1
Logistic Regression	4
Simple Logistic Regression	4
Multiple Logistic Regression – Stepwise selection	5
LDA and QDA	9
LDA	10
QDA	18
kNN	21

Introduction

Libraries

We are going to use:

- **tidyverse**: *Easily Install and Load the 'Tidyverse'*
- **caret**: *Classification and Regression Training*
- **MASS**: *Support Functions and Datasets for Venables and Ripley's MASS*
- **ggplot2**: *Create Elegant Data Visualisations Using the Grammar of Graphics*
- **readxl**: *Read Excel Files*

```
library(tidyverse) # for data manipulation and visualization
library(caret)     # for statistical learning techniques
library(MASS)      # for AIC based stepwise regression
library(ggplot2)   # for plots
library(klaR)       # for LDA and QDA partition
library(readxl)    # for reading xlsx files
```

Data

Today we are going to use the **Titanic** data set.

It contains 1309 rows and 15 columns. Each row represents a passenger and the columns describe some of their attributes.

```
df <- read_excel("Titanic.xlsx")
head(df, 10)
```

```
## # A tibble: 10 x 15
##   pclass survived Residence name    age sibsp parch ticket  fare cabin embarked
##   <dbl>    <dbl>    <dbl> <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr> <chr>
## 1     3        0        0 Abbi~  42     0     0 C.A. ~  7.55 <NA> S
## 2     3        0        0 Abbo~  13     0     2 C.A. ~ 20.2 <NA> S
## 3     3        0        0 Abbo~  16     1     1 C.A. ~ 20.2 <NA> S
## 4     3        1        0 Abbo~  35     1     1 C.A. ~ 20.2 <NA> S
## 5     3        1        2 Abel~  16     0     0 348125  7.65 <NA> S
## 6     3        1        0 Abel~  25     0     0 348122  7.65 F G63 S
## 7     2        0        2 Abel~  30     1     0 P/PP ~ 24    <NA> C
## 8     2        1        2 Abel~  28     1     0 P/PP ~ 24    <NA> C
## 9     3        1        2 Abra~  20     0     0 SOTON~  7.92 <NA> S
## 10    3        1        2 Abra~  18     0     0 2657   7.23 <NA> C
## # ... with 4 more variables: boat <chr>, body <dbl>, home.dest <chr>,
## #   Gender <dbl>
```

Let's see the internal data structure.

```
str(df)
```

```
## tbl_df [1,309 x 15] (S3: tbl_df/tbl/data.frame)
##  $ pclass   : num [1:1309] 3 3 3 3 3 3 2 2 3 3 ...
##  $ survived : num [1:1309] 0 0 0 1 1 1 0 1 1 1 ...
##  $ Residence: num [1:1309] 0 0 0 0 2 0 2 2 2 2 ...
##  $ name      : chr [1:1309] "Abbing, Mr. Anthony" "Abbott, Master. Eugene Joseph" "Abbott, Mr. Rossmo
##  $ age       : num [1:1309] 42 13 16 35 16 25 30 28 20 18 ...
##  $ sibsp     : num [1:1309] 0 0 1 1 0 0 1 1 0 0 ...
##  $ parch     : num [1:1309] 0 2 1 1 0 0 0 0 0 0 ...
##  $ ticket    : chr [1:1309] "C.A. 5547" "C.A. 2673" "C.A. 2673" "C.A. 2673" ...
##  $ fare      : num [1:1309] 7.55 20.25 20.25 20.25 7.65 ...
##  $ cabin     : chr [1:1309] NA NA NA NA ...
##  $ embarked  : chr [1:1309] "S" "S" "S" "S" ...
##  $ boat      : chr [1:1309] NA NA NA "A" ...
##  $ body      : num [1:1309] NA NA 190 NA NA NA NA NA NA ...
##  $ home.dest : chr [1:1309] NA "East Providence, RI" "East Providence, RI" "East Providence, RI" ...
##  $ Gender    : num [1:1309] 0 0 0 1 1 0 0 1 0 1 ...
```

The type of some variables is not the optimal one (e.g., gender as a numeric variable). We can change these through the **mutate** function in **dplyr**.

```
df <- df %>%
  dplyr::mutate(across(c(pclass, survived,      # select features
                        Residence, body, Gender),
                  factor))                    # transform to factor

# this is the traditional approach through base R
# df$pclass   <- as.factor(df$pclass)
# df$survived <- as.factor(df$survived)
# df$Residence <- as.factor(df$Residence)
# df$body     <- as.factor(df$body)
# df$Gender   <- as.factor(df$Gender)
```

According to the column type, the function **summary** provides some summary statistics for each variable.

```
summary(df)
```

```
## pclass survived Residence      name      age
## 1:323    0:809    0:258    Length:1309    Min.   : 0.1667
## 2:277    1:500    1:302    Class :character    1st Qu.:21.0000
## 3:709          2:749    Mode  :character    Median :28.0000
##                                     Mean   :29.8811
##                                     3rd Qu.:39.0000
##                                     Max.   :80.0000
##                                     NA's   :263
##      sibsp      parch      ticket      fare
## Min.   :0.0000    Min.   :0.000    Length:1309    Min.   : 0.000
## 1st Qu.:0.0000    1st Qu.:0.000    Class :character    1st Qu.: 7.896
## Median :0.0000    Median :0.000    Mode  :character    Median : 14.454
## Mean   :0.4989    Mean   :0.385          Mean   : 33.295
## 3rd Qu.:1.0000    3rd Qu.:0.000          3rd Qu.: 31.275
## Max.   :8.0000    Max.   :9.000          Max.   :512.329
##                                     NA's   :1
##      cabin      embarked      boat      body
## Length:1309    Length:1309    Length:1309      1      :    1
## Class :character    Class :character    Class :character    4      :    1
## Mode  :character    Mode  :character    Mode  :character    7      :    1
##                                     9      :    1
##                                     14     :    1
##                                     (Other): 116
##                                     NA's   :1188
##      home.dest      Gender
## Length:1309      0:843
## Class :character  1:466
## Mode  :character
##
##
##
```

Now, we drop the columns having more than 50 percent of NAs, as well as the ones that will not be useful for our analysis.

We also drop all those rows without age information.

```
# selecting columns
df <- df %>%
  dplyr::select(-c(name,ticket, fare, cabin,
                    embarked, boat, body, home.dest))

# filtering out rows
df <- df %>% filter(!is.na(age))

# our "final" dataset
dim(df)
```

```
## [1] 1046      7
```

```
summary(df)
```

```
## pclass survived Residence      age      sibsp      parch
```

```
## 1:284 0:619 0:232 Min. : 0.1667 Min. :0.0000 Min. :0.0000
## 2:261 1:427 1:248 1st Qu.:21.0000 1st Qu.:0.0000 1st Qu.:0.0000
## 3:501 2:566 Median :28.0000 Median :0.0000 Median :0.0000
## Mean :29.8811 Mean :0.5029 Mean :0.4207
## 3rd Qu.:39.0000 3rd Qu.:1.0000 3rd Qu.:1.0000
## Max. :80.0000 Max. :8.0000 Max. :6.0000
## Gender
## 0:658
## 1:388
##
##
##
##
```

Logistic Regression

To assess prediction accuracy, we split the data into training and testing sets (at random, with no replacement). These encompass 75 and 25 percent of the points, respectively.

```
set.seed(123)
# set.seed(1) # try to re-run the analysis with this
training_samples <- df$survived %>%
  caret::createDataPartition(p = 0.75, list = FALSE)

train <- df[training_samples, ]
test <- df[-training_samples, ]
```

Simple Logistic Regression

Using the training set, we build a simple logistic regression model using sex as the only explanatory variable of the survival status for each passenger.

```
simple_glm <- glm(survived ~ Gender,
  data = train, family = 'binomial')
summary(simple_glm)
```

```
##
## Call:
## glm(formula = survived ~ Gender, family = "binomial", data = train)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -1.6274  -0.6878  -0.6878   0.7864   1.7650
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.3211     0.1109  -11.91  <2e-16 ***
## Gender1       2.3362     0.1719   13.59  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 1063.10  on 785  degrees of freedom
## Residual deviance:  847.53  on 784  degrees of freedom
```

```
## AIC: 851.53
##
## Number of Fisher Scoring iterations: 4
```

You can extract various information from the output object `simple_glm`, such as the regression coefficients:

```
simple_glm$coefficients
```

```
## (Intercept)      Gender1
##   -1.321108      2.336156
```

Our model is $\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 \text{Gender}$, where $\beta_0 = \text{Intercept}$ and $\beta_1 = \text{Gender1}$. Therefore, $\log\left(\frac{p(x)}{1-p(x)}\right) = \text{Intercept} + \text{Gender1} \times \text{Gender}$.

Let's see predictive power of our model in terms of *accuracy* – i.e. the proportion of correct predictions, both true positives and true negatives, among the total number of cases examined.

Since our response (Survival) is a binary variable, we need to round the probabilities which are predicted by the logistic model.

```
# Test for accuracy: predict test data
predict_sex_survived <- predict(simple_glm, newdata = test, type = 'response')

# round up the predictions
predict_sex_survived <- ifelse(predict_sex_survived > 0.5, 1, 0)

# calculate accuracy
accuracyRed <- mean(predict_sex_survived == test$survived)
accuracyRed
```

```
## [1] 0.8115385
```

Our result is “pretty good” and it is also consistent with the “women and children first” code of conduct/policy (you can find more details on [Wikipedia](#)).

Multiple Logistic Regression – Stepwise selection

We may try to improve predictive power by including more features into the model. Let us start with the full/saturated model (i.e., the one comprising all features).

```
glm_complete <- glm(survived ~ ., data=train, family = 'binomial')
summary(glm_complete)
```

```
##
## Call:
## glm(formula = survived ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.7502  -0.6744  -0.4067   0.6634   2.5247
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.638344   0.390539   4.195 2.73e-05 ***
## pclass2     -1.177714   0.291066  -4.046 5.21e-05 ***
## pclass3     -2.372486   0.296672  -7.997 1.27e-15 ***
## Residence1  -0.483365   0.307982  -1.569  0.11654
## Residence2  -0.125636   0.271809  -0.462  0.64392
## age         -0.044028   0.007677  -5.735 9.74e-09 ***
```

```
## sibsp      -0.384458   0.118970  -3.232   0.00123 **
## parch      0.145025   0.109662   1.322   0.18601
## Gender1    2.428404   0.199612  12.166  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 1063.10  on 785  degrees of freedom
## Residual deviance:  729.57  on 777  degrees of freedom
## AIC: 747.57
##
## Number of Fisher Scoring iterations: 5
```

Let's see its predictive accuracy. Has it improved?

```
# Test for accuracy: predict test data
predict_sex_survived <- predict(glm_complete, newdata = test, type = 'response')

# round up the predictions
predict_sex_survived <- ifelse(predict_sex_survived > 0.5, 1, 0)

# calculate accuracy
accuracySat <- mean(predict_sex_survived == test$survived)
accuracySat
```

```
## [1] 0.8
```

```
confusionMatrix(as.factor(predict_sex_survived), test$survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0    1
##           0 132  30
##           1  22  76
##
##           Accuracy : 0.8
##           95% CI : (0.7461, 0.8469)
##      No Information Rate : 0.5923
##      P-Value [Acc > NIR] : 8.187e-13
##
##           Kappa : 0.581
##
##  Mcnemar's Test P-Value : 0.3317
##
##           Sensitivity : 0.8571
##           Specificity : 0.7170
##           Pos Pred Value : 0.8148
##           Neg Pred Value : 0.7755
##           Prevalence : 0.5923
##           Detection Rate : 0.5077
##      Detection Prevalence : 0.6231
##           Balanced Accuracy : 0.7871
##
##           'Positive' Class : 0
```

```
##
```

Moreover, we notice that some features result as non-significant according to their p -values (last column of the previous table). We can select “relevant” predictors through stepwise regression, and then compute an information criterion such as AIC to compare different sub-models and pick the “best” one (e.g. minimizing the AIC).

```
glm_stepwise <- glm_complete %>%  
  MASS::stepAIC(direction='both', trace = T)
```

```
## Start: AIC=747.57
```

```
## survived ~ pclass + Residence + age + sibsp + parch + Gender
```

```
##
```

	Df	Deviance	AIC
## - Residence	2	732.48	746.48
## - parch	1	731.31	747.31
## <none>		729.57	747.57
## - sibsp	1	741.04	757.04
## - age	1	766.10	782.10
## - pclass	2	804.71	818.71
## - Gender	1	907.90	923.90

```
##
```

```
## Step: AIC=746.48
```

```
## survived ~ pclass + age + sibsp + parch + Gender
```

```
##
```

	Df	Deviance	AIC
## - parch	1	734.08	746.08
## <none>		732.48	746.48
## + Residence	2	729.57	747.57
## - sibsp	1	744.58	756.58
## - age	1	770.62	782.62
## - pclass	2	836.37	846.37
## - Gender	1	916.05	928.05

```
##
```

```
## Step: AIC=746.08
```

```
## survived ~ pclass + age + sibsp + Gender
```

```
##
```

	Df	Deviance	AIC
## <none>		734.08	746.08
## + parch	1	732.48	746.48
## + Residence	2	731.31	747.31
## - sibsp	1	744.61	754.61
## - age	1	773.15	783.15
## - pclass	2	837.71	845.71
## - Gender	1	932.11	942.11

```
summary(glm_stepwise)
```

```
##
```

```
## Call:
```

```
## glm(formula = survived ~ pclass + age + sibsp + Gender, family = "binomial",  
##      data = train)
```

```
##
```

```
## Deviance Residuals:
```

##	Min	1Q	Median	3Q	Max
##	-2.7215	-0.6664	-0.4103	0.6854	2.5462

```
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.619044   0.364356   4.444 8.85e-06 ***
## pclass2      -1.383870   0.263438  -5.253 1.50e-07 ***
## pclass3      -2.469957   0.264871  -9.325 < 2e-16 ***
## age          -0.045191   0.007606  -5.942 2.82e-09 ***
## sibsp        -0.346727   0.111916  -3.098 0.00195 **
## Gender1       2.496307   0.196409  12.710 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##    Null deviance: 1063.10  on 785  degrees of freedom
## Residual deviance:  734.08  on 780  degrees of freedom
## AIC: 746.08
##
## Number of Fisher Scoring iterations: 4
```

A comparison of the simple, saturated and the selected model using AIC indicate that the latter is a good compromise between the two. It should have a “comparable” predictive power but a lower complexity compared to the saturated one.

```
AIC(simple_glm, glm_complete, glm_stepwise)
```

```
##              df      AIC
## simple_glm    2 851.5342
## glm_complete  9 747.5722
## glm_stepwise  6 746.0796
```

Using the selected model, we compute again the probability for survival on the test set, and the overall prediction accuracy.

```
predict_sex_survived <- predict(glm_stepwise, newdata = test, type = 'response')

predict_sex_survived <- ifelse(predict_sex_survived > 0.5, 1, 0)

accuracy <- mean(predict_sex_survived == test$survived)
accuracy
```

```
## [1] 0.8038462
```

We can create also a confusion matrix (and statistics) to compare our predictions for the test set.

```
confusionMatrix(as.factor(predict_sex_survived), test$survived)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    0    1
##              0 133  30
##              1  21  76
##
##              Accuracy : 0.8038
##              95% CI : (0.7503, 0.8503)
##    No Information Rate : 0.5923
##    P-Value [Acc > NIR] : 2.921e-13
```



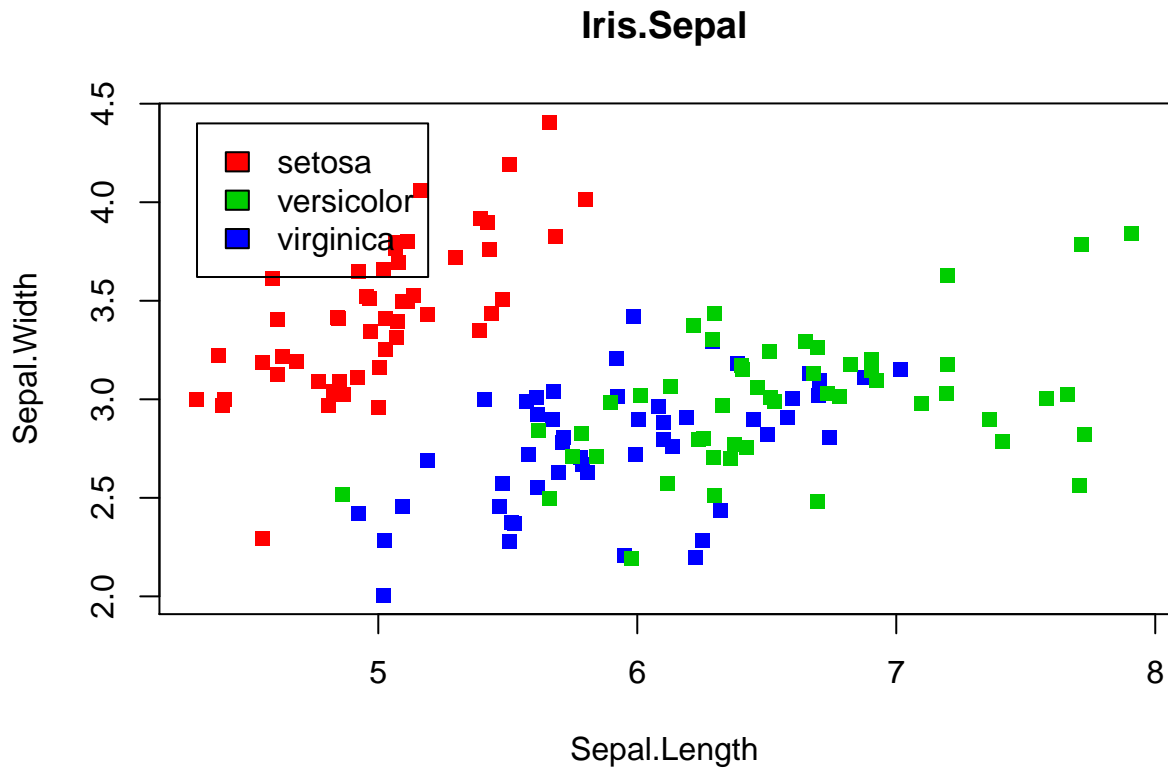
```
##
##           Kappa : 0.5884
##
## Mcnemar's Test P-Value : 0.2626
##
##           Sensitivity : 0.8636
##           Specificity : 0.7170
##           Pos Pred Value : 0.8160
##           Neg Pred Value : 0.7835
##           Prevalence : 0.5923
##           Detection Rate : 0.5115
##           Detection Prevalence : 0.6269
##           Balanced Accuracy : 0.7903
##
##           'Positive' Class : 0
##
```

LDA and QDA

Let us apply LDA and QDA to a multi label dataset such as **iris**. We are going to use just the first two columns with a gaussian noise.

```
iris2 <- iris[,c(1,2,5)]
species_name <- iris$Species
iris2[,1] <- iris2[,1] + rnorm(150, sd=0.025)
iris2[,2] <- iris2[,2] + rnorm(150, sd=0.025)

plot(iris2[,1:2], main='Iris.Sepal', xlab='Sepal.Length', ylab='Sepal.Width', pch=15)
points(iris2[1:50,], col=2, pch=15)
points(iris2[51:100,], col=4, pch=15)
points(iris2[101:150,], col=3, pch=15)
legend(min(iris[,1]), max(iris[,2]), legend=levels(species_name), fill=c(2,3,4))
```



Once again we create a train set and a test set.

```
set.seed(123)
training.samples <- species_name %>%
  createDataPartition(p = 0.8, list = FALSE)
train <- iris2[training.samples, ]
test <- iris2[-training.samples, ]
```

It is generally recommended to standardize/normalize continuous predictor before the analysis.

```
help(preProcess)

# Estimate preprocessing parameters
preproc.param <- train %>%
  preProcess(method = c("center", "scale"))
# Transform the data using the estimated parameters
train_transformed <- preproc.param %>% predict(train)
test_transformed <- preproc.param %>% predict(test)
```

LDA

Before performing LDA, consider:

- Inspecting the univariate distributions of each variable and check whether they are normally distributed. If not, you can transform them using log and root for exponential distributions and Box-Cox for skewed distributions.
- Standardize the variables to make their scale comparable.

- Be careful for the possible presence of outliers, and remember: “*any reasonable, formal or informal, procedure for rejecting outliers will prevent the worst*” (P.J. Huber).

```
help(lda)

lda.iris <- lda(factor(Species)~ Sepal.Length + Sepal.Width, data=train_transformed)
lda.iris

## Call:
## lda(factor(Species) ~ Sepal.Length + Sepal.Width, data = train_transformed)
##
## Prior probabilities of groups:
##      setosa versicolor virginica
## 0.3333333 0.3333333 0.3333333
##
## Group means:
##      Sepal.Length Sepal.Width
## setosa      -1.0120940  0.77319367
## versicolor   0.1004989 -0.68484625
## virginica    0.9115952 -0.08834742
##
## Coefficients of linear discriminants:
##      LD1      LD2
## Sepal.Length -1.775634 -0.6120093
## Sepal.Width  1.086270 -0.9378600
##
## Proportion of trace:
##      LD1      LD2
## 0.9493 0.0507
```

In the output above, aside from prior provability and group means, we have:

- *Coefficients of linear discriminants*: the linear combination of predictor variables that are used to form the LDA decision rule.
- *Proportion of trace*: the separation achieved by each discriminant function (in percentage).

Here is the model accuracy on training data.

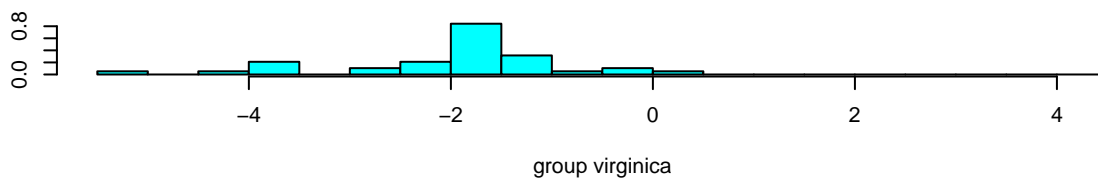
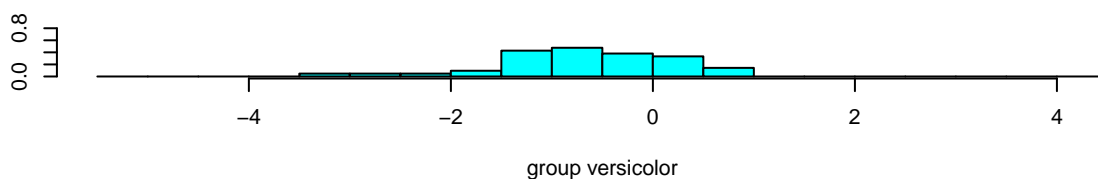
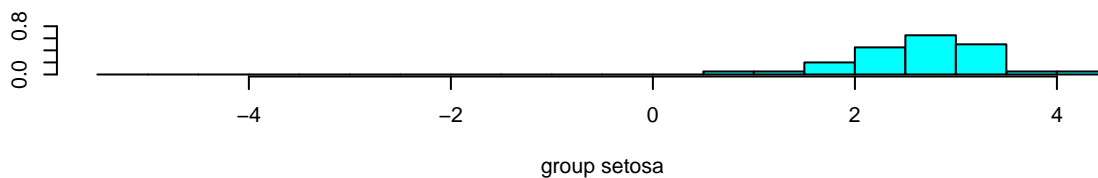
```
predmodel.train.lda = predict(lda.iris, data=train_transformed)
confusionMatrix(as.factor(predmodel.train.lda$class), train_transformed$Species)

## Confusion Matrix and Statistics
##
##              Reference
## Prediction  setosa versicolor virginica
## setosa      39          1          0
## versicolor   1         29         12
## virginica    0         10         28
##
## Overall Statistics
##
##              Accuracy : 0.8
##              95% CI : (0.7172, 0.8675)
##      No Information Rate : 0.3333
##      P-Value [Acc > NIR] : < 2.2e-16
##
##              Kappa : 0.7
```

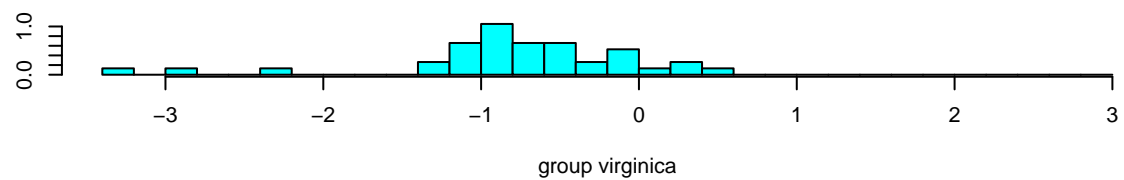
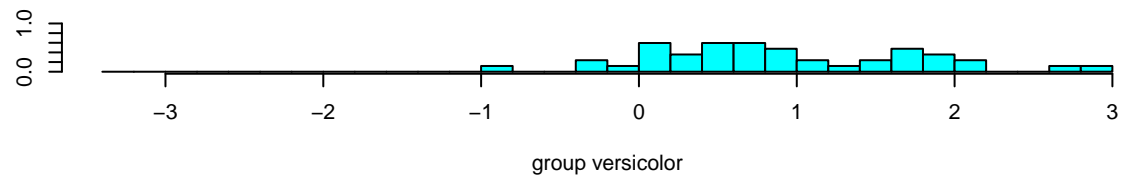
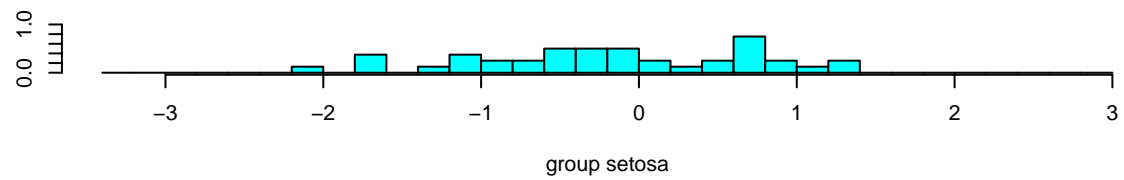
```
##
## McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##          Class: setosa Class: versicolor Class: virginica
## Sensitivity          0.9750          0.7250          0.7000
## Specificity          0.9875          0.8375          0.8750
## Pos Pred Value       0.9750          0.6905          0.7368
## Neg Pred Value       0.9875          0.8590          0.8537
## Prevalence           0.3333          0.3333          0.3333
## Detection Rate       0.3250          0.2417          0.2333
## Detection Prevalence 0.3333          0.3500          0.3167
## Balanced Accuracy    0.9812          0.7812          0.7875
```

The plot below shows how the response class has been classified by the LDA classifier. The x -axis shows the value of the line defined by the coefficient of linear discriminant for LDA. Groups are the ones in the response classes.

```
# first discriminant
ldahist(predmodel.train.lda$x[,1], g= predmodel.train.lda$class)
```

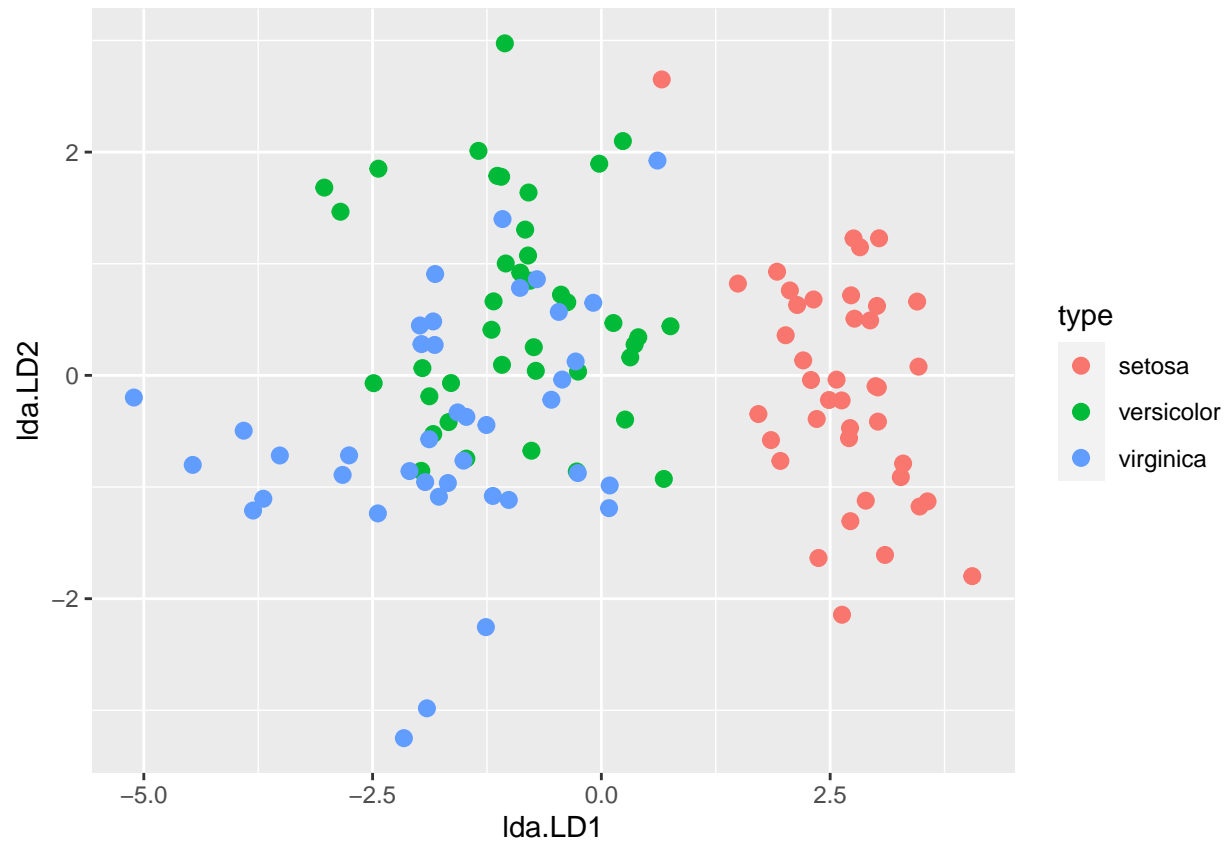


```
# second discriminant
ldahist(predmodel.train.lda$x[,2], g= predmodel.train.lda$class)
```



See new x with original labels

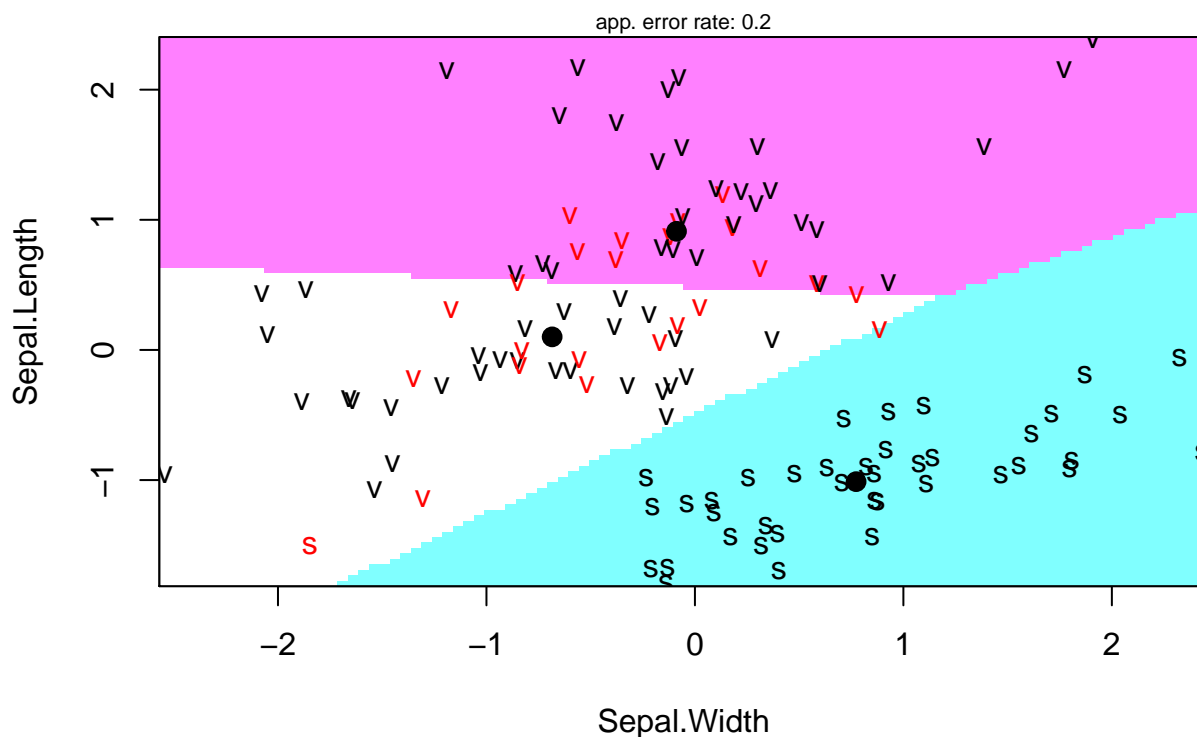
```
#convert to data frame
newdata <- data.frame(type = train_transformed$Species, lda = predmodel.train.lda$x)
library(ggplot2)
ggplot(newdata) + geom_point(aes(lda.LD1, lda.LD2, colour = type), size = 2.5)
```



See geometric division

```
# library(klaR)
partimat(factor(Species) ~ Sepal.Length + Sepal.Width,
  data=train_transformed, method = "lda")
```

Partition Plot



Now we check the model accuracy on test data.

```
predmodel.test.lda = predict(lda.iris, newdata=test_transformed)
confusionMatrix(as.factor(predmodel.test.lda$class), test_transformed$Species)
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##           Reference
```

```
## Prediction  setosa versicolor virginica
```

```
##   setosa      10         0         0
```

```
## versicolor   0         8         4
```

```
## virginica    0         2         6
```

```
##
```

```
## Overall Statistics
```

```
##
```

```
##           Accuracy : 0.8
```

```
##           95% CI : (0.6143, 0.9229)
```

```
## No Information Rate : 0.3333
```

```
## P-Value [Acc > NIR] : 2.09e-07
```

```
##
```

```
##           Kappa : 0.7
```

```
##
```

```
## McNemar's Test P-Value : NA
```

```
##
```

```
## Statistics by Class:
```

```
##
```

```
##           Class: setosa Class: versicolor Class: virginica
```

## Sensitivity	1.0000	0.8000	0.6000
## Specificity	1.0000	0.8000	0.9000
## Pos Pred Value	1.0000	0.6667	0.7500
## Neg Pred Value	1.0000	0.8889	0.8182
## Prevalence	0.3333	0.3333	0.3333
## Detection Rate	0.3333	0.2667	0.2000
## Detection Prevalence	0.3333	0.4000	0.2667
## Balanced Accuracy	1.0000	0.8000	0.7500

Let's try LDA with all variables in iris data:

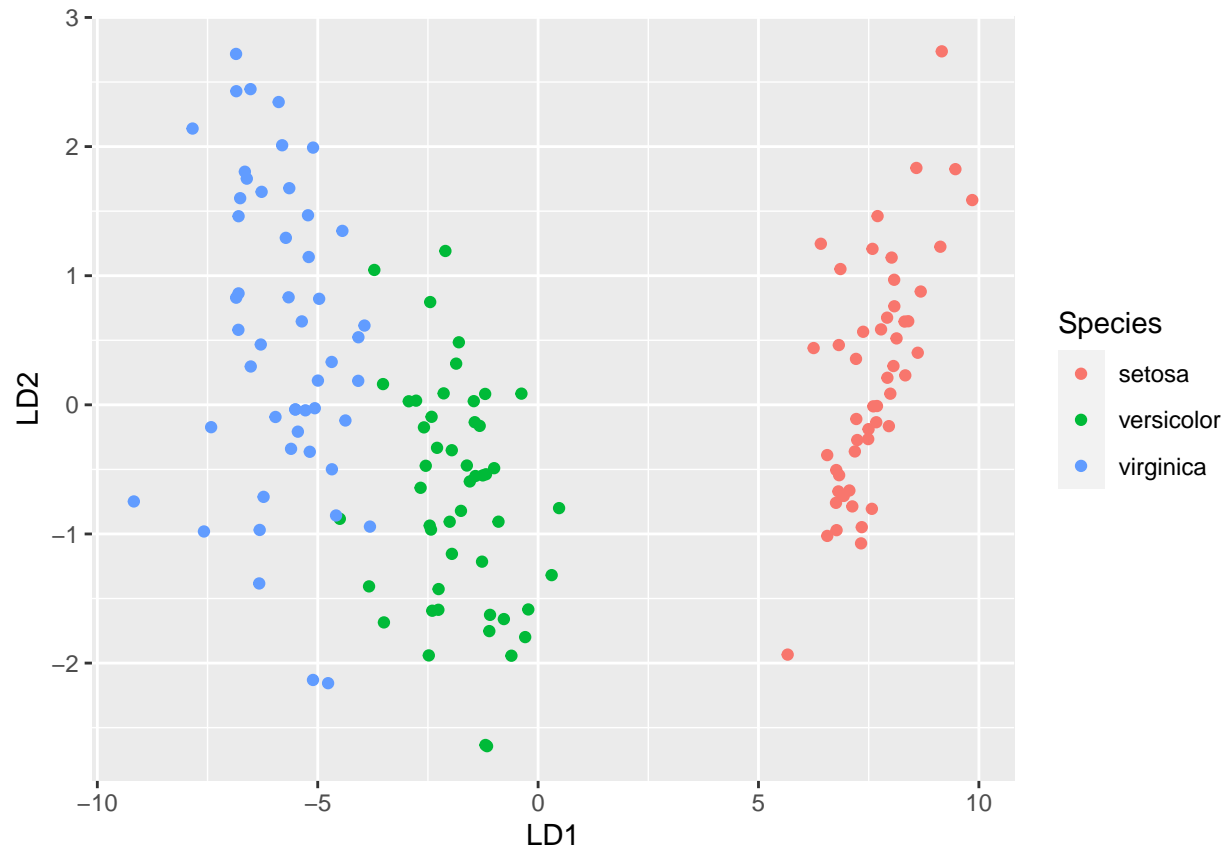
```
irisLda <- lda(Species~.,data=iris)
irisLda
```

```
## Call:
## lda(Species ~ ., data = iris)
##
## Prior probabilities of groups:
##      setosa versicolor  virginica
## 0.3333333 0.3333333 0.3333333
##
## Group means:
##           Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa           5.006      3.428         1.462      0.246
## versicolor       5.936      2.770         4.260      1.326
## virginica        6.588      2.974         5.552      2.026
##
## Coefficients of linear discriminants:
##           LD1          LD2
## Sepal.Length 0.8293776 0.02410215
## Sepal.Width  1.5344731 2.16452123
## Petal.Length -2.2012117 -0.93192121
## Petal.Width  -2.8104603 2.83918785
##
## Proportion of trace:
##      LD1      LD2
## 0.9912 0.0088
```

```
scalIris <- scale(as.matrix(iris[,-5]),scale=FALSE)
```

```
irisProjection <- cbind(scalIris %*% irisLda$scaling, iris[,5,drop=FALSE])
```

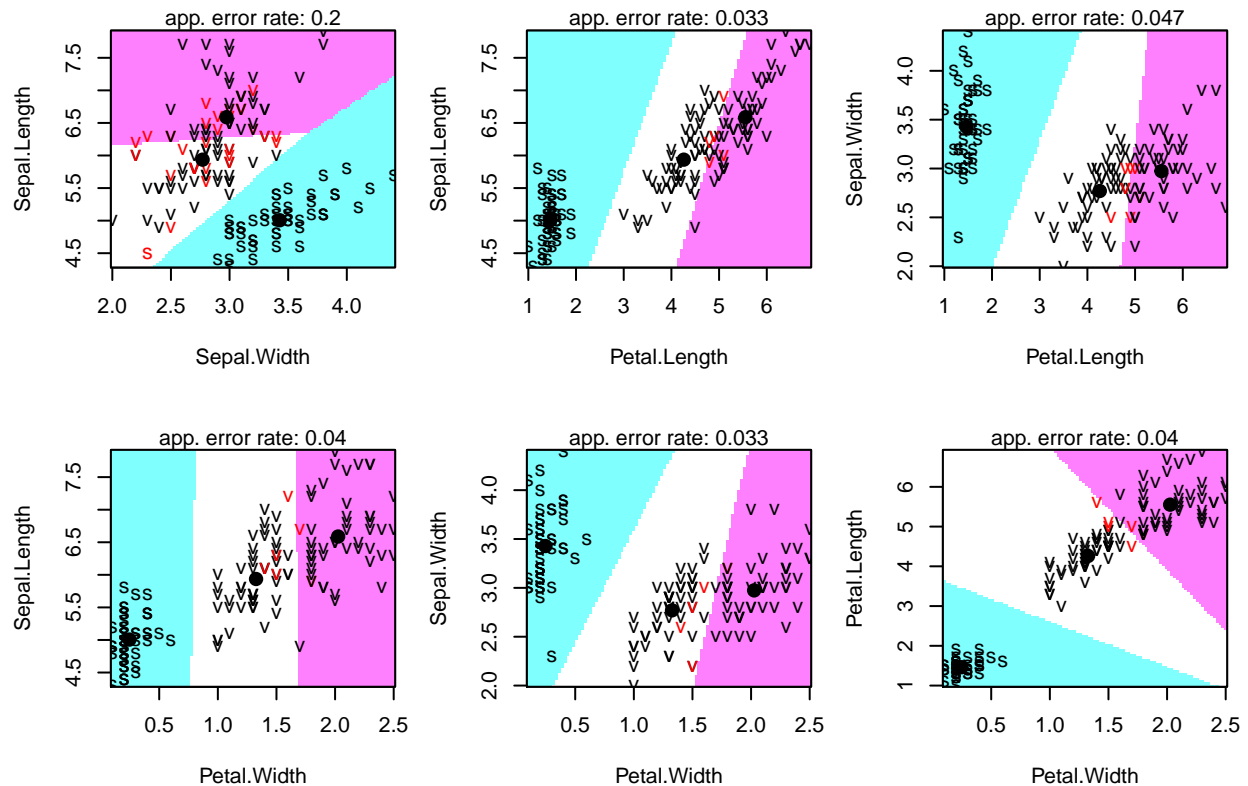
```
p <- ggplot(data=irisProjection,aes(x=LD1,y=LD2,col=Species))
p + geom_point()
```

Geometric division

```
partimat(factor(Species)~.,  
  data=iris, method = "lda")
```

Partition Plot



QDA

Next we will fit the model through QDA. The command is similar to LDA and it outputs the prior probabilities and Group means. Note that “Prior Probabilities” and “Group Means” values are the same as of LDA.

```
qda.iris <- qda(factor(Species) ~ Sepal.Length + Sepal.Width, data=train_transformed)
qda.iris
```

```
## Call:
## qda(factor(Species) ~ Sepal.Length + Sepal.Width, data = train_transformed)
##
## Prior probabilities of groups:
##      setosa versicolor virginica
## 0.3333333 0.3333333 0.3333333
##
## Group means:
##           Sepal.Length Sepal.Width
## setosa      -1.0120940  0.77319367
## versicolor   0.1004989 -0.68484625
## virginica    0.9115952 -0.08834742
```

We will find the model accuracy for training data.

```
predmodel.train.qda = predict(qda.iris, data=train_transformed)
confusionMatrix(as.factor(predmodel.train.qda$class), train_transformed$Species)
```

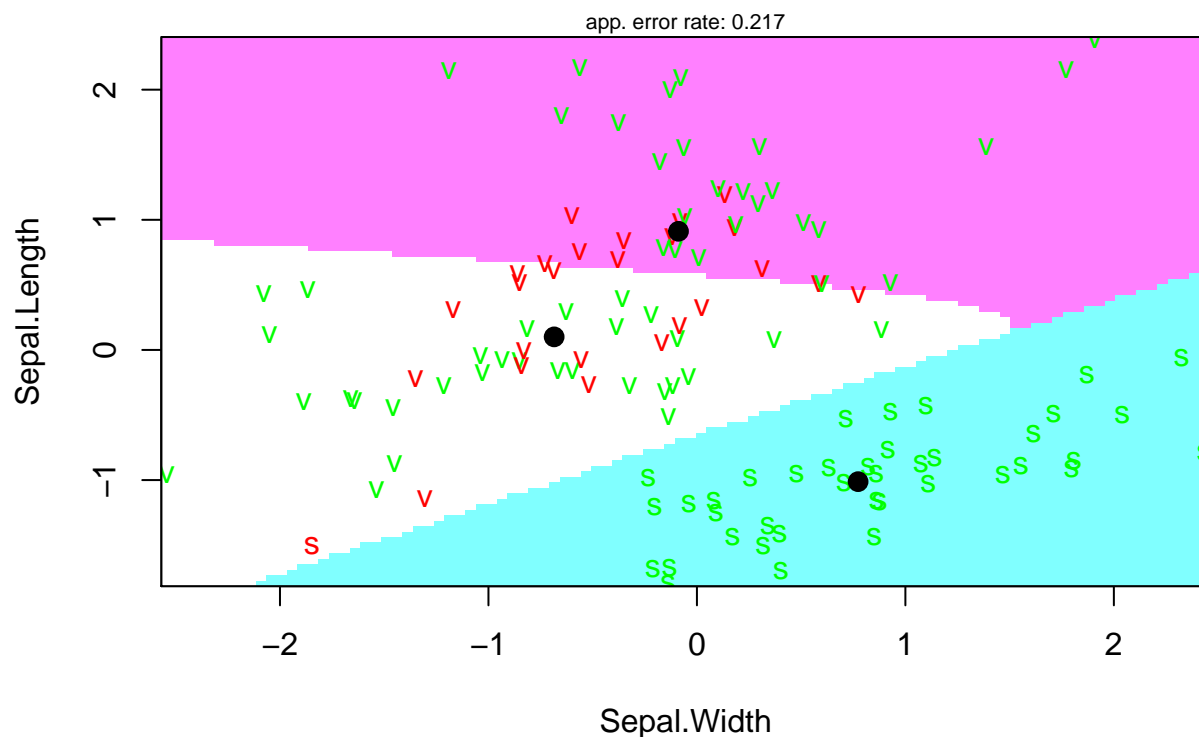
```
## Confusion Matrix and Statistics
##
```

```
##               Reference
## Prediction   setosa versicolor virginica
##   setosa      39         0         0
##   versicolor  1         30        15
##   virginica   0         10        25
##
## Overall Statistics
##
##               Accuracy : 0.7833
##               95% CI : (0.6989, 0.8533)
##   No Information Rate : 0.3333
##   P-Value [Acc > NIR] : < 2.2e-16
##
##               Kappa : 0.675
##
## McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##               Class: setosa Class: versicolor Class: virginica
## Sensitivity      0.9750      0.7500      0.6250
## Specificity      1.0000      0.8000      0.8750
## Pos Pred Value   1.0000      0.6522      0.7143
## Neg Pred Value   0.9877      0.8649      0.8235
## Prevalence       0.3333      0.3333      0.3333
## Detection Rate   0.3250      0.2500      0.2083
## Detection Prevalence 0.3250      0.3833      0.2917
## Balanced Accuracy 0.9875      0.7750      0.7500
```

We can see the geometric partition

```
# library(klaR)
partimat(factor(Species) ~ Sepal.Length + Sepal.Width,
  data=train_transformed, method = "qda",
  col.correct='green', col.wrong='red')
```

Partition Plot



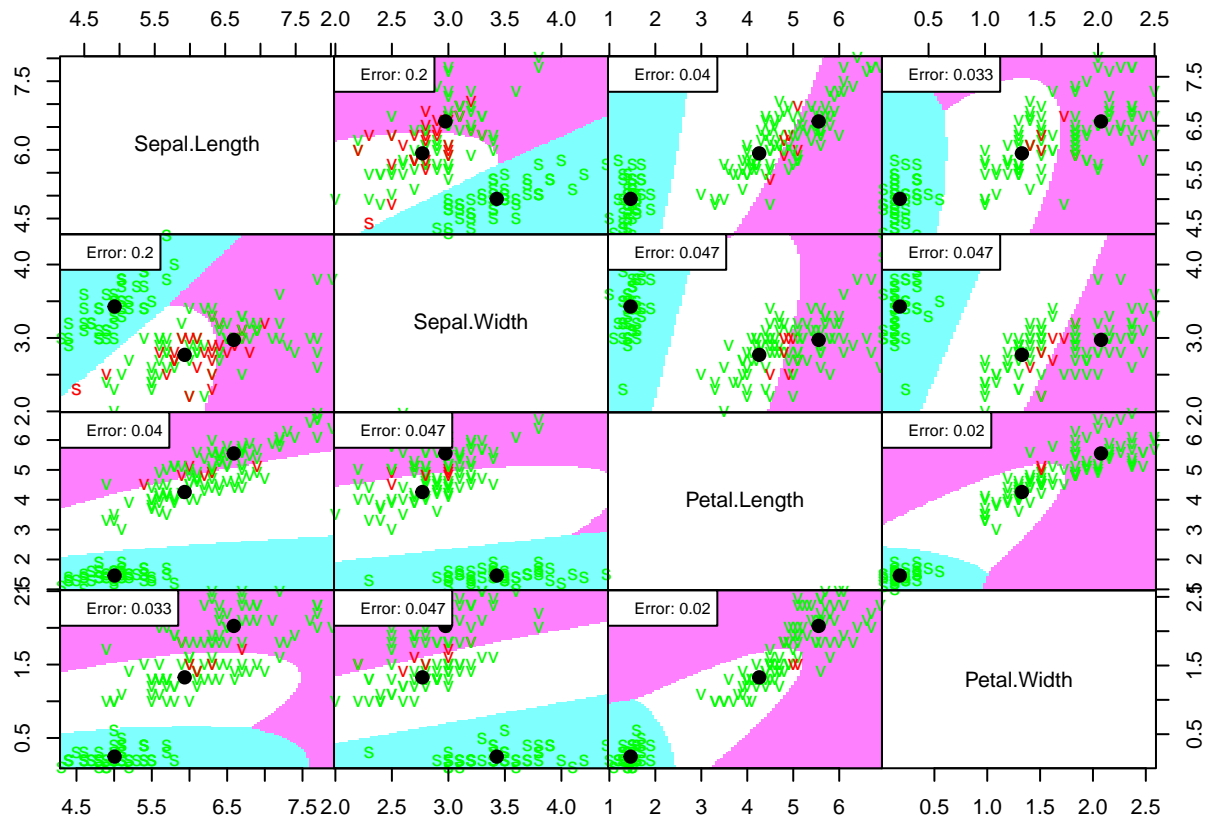
Let's also try QDA on all original variables

```
irisQda <- lda(Species ~ ., data = iris)
irisQda
```

```
## Call:
## lda(Species ~ ., data = iris)
##
## Prior probabilities of groups:
##      setosa versicolor  virginica
## 0.3333333 0.3333333 0.3333333
##
## Group means:
##      Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa           5.006      3.428       1.462      0.246
## versicolor       5.936      2.770       4.260      1.326
## virginica        6.588      2.974       5.552      2.026
##
## Coefficients of linear discriminants:
##      LD1      LD2
## Sepal.Length 0.8293776 0.02410215
## Sepal.Width  1.5344731 2.16452123
## Petal.Length -2.2012117 -0.93192121
## Petal.Width  -2.8104603 2.83918785
##
## Proportion of trace:
##      LD1      LD2
```

```
## 0.9912 0.0088
```

```
partimat(Species ~ ., data = iris, method = "qda",
  plot.matrix = TRUE, col.correct='green', col.wrong='red')
```



k NN

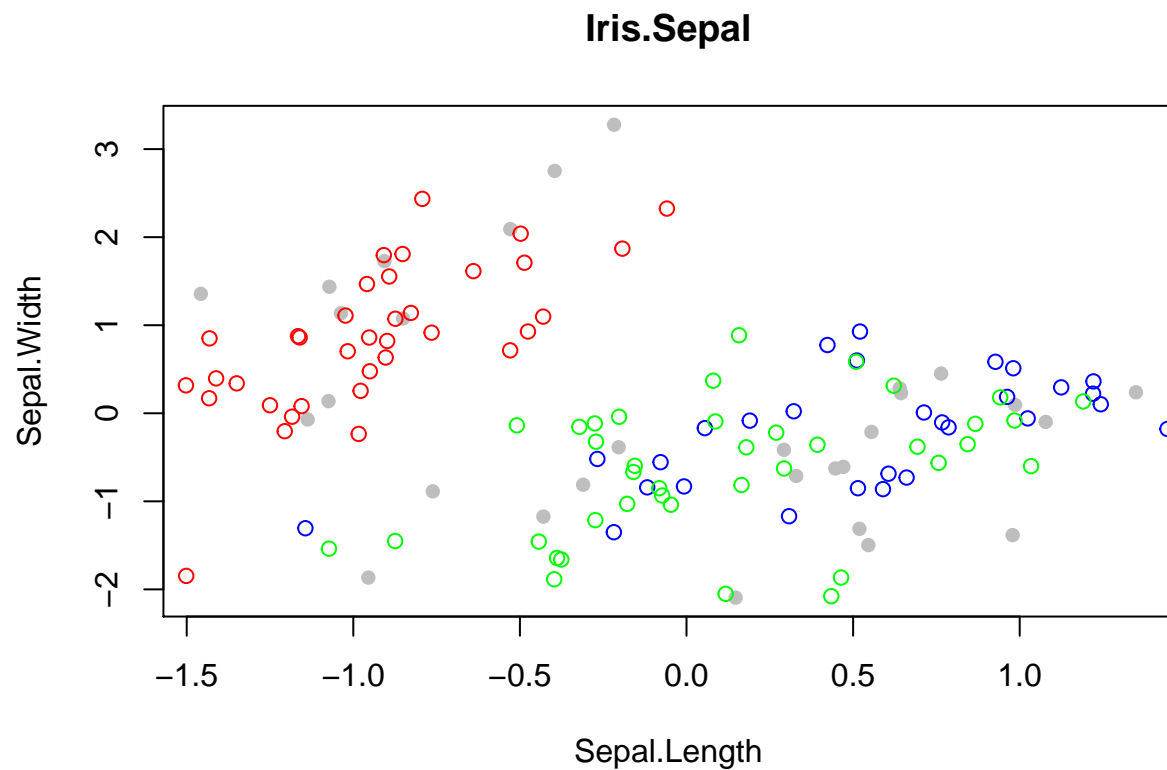
We are going to use the `knn3()` function within `caret` package on the `iris2` data (i.e. working on 2 dimensions). Let's train the knn with $k = 1, 2, 3, 10$

```
knn_iris1 <- knn3(factor(Species) ~ Sepal.Length + Sepal.Width,
  data=train_transformed, k = 1)
knn_iris2 <- knn3(factor(Species) ~ Sepal.Length + Sepal.Width,
  data=train_transformed, k = 2)
knn_iris3 <- knn3(factor(Species) ~ Sepal.Length + Sepal.Width,
  data=train_transformed, k = 3)
knn_iris10 <- knn3(factor(Species) ~ Sepal.Length + Sepal.Width,
  data=train_transformed, k = 10)
```

Hand-made KNN:

```
plot(test_transformed[,1:2], main='Iris.Sepal',
  xlab='Sepal.Length', ylab='Sepal.Width', pch=16, col='grey')
points(train_transformed[which(train_transformed$Species=="setosa"),1:2],
  col='red', pch=1)
points(train_transformed[which(train_transformed$Species=="virginica"),1:2],
  col='blue', pch=1)
```

```
points(train_transformed[which(train_transformed$Species=="versicolor"),1:2],
       col='green', pch=1)
```



And now let us predict new points labels in the test set. Using $k = 1$:

```
predict(knn_iris1, test_transformed, type='prob')
```

##	setosa	versicolor	virginica
## [1,]	1	0	0
## [2,]	1	0	0
## [3,]	1	0	0
## [4,]	1	0	0
## [5,]	1	0	0
## [6,]	1	0	0
## [7,]	1	0	0
## [8,]	1	0	0
## [9,]	1	0	0
## [10,]	1	0	0
## [11,]	0	0	1
## [12,]	0	0	1
## [13,]	0	1	0
## [14,]	0	1	0
## [15,]	0	1	0
## [16,]	0	0	1
## [17,]	0	1	0
## [18,]	0	1	0
## [19,]	0	0	1

```
## [20,]    0      1      0
## [21,]    0      1      0
## [22,]    0      0      1
## [23,]    0      1      0
## [24,]    0      0      1
## [25,]    0      1      0
## [26,]    0      1      0
## [27,]    0      1      0
## [28,]    0      0      1
## [29,]    0      1      0
## [30,]    0      0      1
```

```
predict_test_knn1 <- predict(knn_iris1, test_transformed, type='class')
confusionMatrix(predict_test_knn1, test_transformed$Species)
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##           Reference
```

```
## Prediction  setosa versicolor virginica
```

```
##   setosa      10          0          0
```

```
##   versicolor  0          6          6
```

```
##   virginica   0          4          4
```

```
##
```

```
## Overall Statistics
```

```
##
```

```
##           Accuracy : 0.6667
```

```
##           95% CI : (0.4719, 0.8271)
```

```
##   No Information Rate : 0.3333
```

```
##   P-Value [Acc > NIR] : 0.0001938
```

```
##
```

```
##           Kappa : 0.5
```

```
##
```

```
##   McNemar's Test P-Value : NA
```

```
##
```

```
## Statistics by Class:
```

```
##
```

```
##           Class: setosa Class: versicolor Class: virginica
```

```
## Sensitivity           1.0000           0.6000           0.4000
```

```
## Specificity           1.0000           0.7000           0.8000
```

```
## Pos Pred Value        1.0000           0.5000           0.5000
```

```
## Neg Pred Value        1.0000           0.7778           0.7273
```

```
## Prevalence            0.3333           0.3333           0.3333
```

```
## Detection Rate        0.3333           0.2000           0.1333
```

```
## Detection Prevalence  0.3333           0.4000           0.2667
```

```
## Balanced Accuracy      1.0000           0.6500           0.6000
```

```
Using  $k = 2$ :
```

```
predict(knn_iris2, test_transformed, type='prob')
```

```
##           setosa versicolor virginica
```

```
## [1,]      1      0.0      0.0
```

```
## [2,]      1      0.0      0.0
```

```
## [3,]      1      0.0      0.0
```

```
## [4,]      1      0.0      0.0
```

```
## [5,]      1      0.0      0.0
```

```
## [6,]      1      0.0      0.0
## [7,]      1      0.0      0.0
## [8,]      1      0.0      0.0
## [9,]      1      0.0      0.0
## [10,]     1      0.0      0.0
## [11,]     0      0.0      1.0
## [12,]     0      0.5      0.5
## [13,]     0      1.0      0.0
## [14,]     0      0.5      0.5
## [15,]     0      1.0      0.0
## [16,]     0      0.5      0.5
## [17,]     0      0.5      0.5
## [18,]     0      1.0      0.0
## [19,]     0      0.5      0.5
## [20,]     0      0.5      0.5
## [21,]     0      1.0      0.0
## [22,]     0      0.0      1.0
## [23,]     0      0.5      0.5
## [24,]     0      0.5      0.5
## [25,]     0      0.5      0.5
## [26,]     0      1.0      0.0
## [27,]     0      0.5      0.5
## [28,]     0      0.5      0.5
## [29,]     0      0.5      0.5
## [30,]     0      0.0      1.0
```

```
predict_test_knn2 <- predict(knn_iris2, test_transformed, type='class')
confusionMatrix(predict_test_knn2, test_transformed$Species)
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##           Reference
```

```
## Prediction  setosa versicolor virginica
```

```
##   setosa      10         0         0
```

```
## versicolor   0         5         5
```

```
## virginica    0         5         5
```

```
##
```

```
## Overall Statistics
```

```
##
```

```
##           Accuracy : 0.6667
```

```
##           95% CI : (0.4719, 0.8271)
```

```
## No Information Rate : 0.3333
```

```
## P-Value [Acc > NIR] : 0.0001938
```

```
##
```

```
##           Kappa : 0.5
```

```
##
```

```
## McNemar's Test P-Value : NA
```

```
##
```

```
## Statistics by Class:
```

```
##
```

```
##           Class: setosa Class: versicolor Class: virginica
```

```
## Sensitivity           1.0000           0.5000           0.5000
```

```
## Specificity           1.0000           0.7500           0.7500
```

```
## Pos Pred Value        1.0000           0.5000           0.5000
```

```
## Neg Pred Value        1.0000           0.7500           0.7500
```


## Prevalence	0.3333	0.3333	0.3333
## Detection Rate	0.3333	0.1667	0.1667
## Detection Prevalence	0.3333	0.3333	0.3333
## Balanced Accuracy	1.0000	0.6250	0.6250

Using $k = 3$:

```
# k=3
predict(knn_iris3, test_transformed, type='prob')
```

```
##          setosa versicolor virginica
## [1,] 1.0000000 0.0000000 0.0000000
## [2,] 1.0000000 0.0000000 0.0000000
## [3,] 1.0000000 0.0000000 0.0000000
## [4,] 1.0000000 0.0000000 0.0000000
## [5,] 1.0000000 0.0000000 0.0000000
## [6,] 1.0000000 0.0000000 0.0000000
## [7,] 1.0000000 0.0000000 0.0000000
## [8,] 1.0000000 0.0000000 0.0000000
## [9,] 1.0000000 0.0000000 0.0000000
## [10,] 1.0000000 0.0000000 0.0000000
## [11,] 0.0000000 0.0000000 1.0000000
## [12,] 0.0000000 0.6666667 0.3333333
## [13,] 0.0000000 1.0000000 0.0000000
## [14,] 0.0000000 0.6666667 0.3333333
## [15,] 0.0000000 0.6666667 0.3333333
## [16,] 0.0000000 0.3333333 0.6666667
## [17,] 0.0000000 0.6666667 0.3333333
## [18,] 0.3333333 0.6666667 0.0000000
## [19,] 0.0000000 0.6666667 0.3333333
## [20,] 0.0000000 0.3333333 0.6666667
## [21,] 0.0000000 0.6666667 0.3333333
## [22,] 0.0000000 0.3333333 0.6666667
## [23,] 0.0000000 0.3333333 0.6666667
## [24,] 0.0000000 0.6666667 0.3333333
## [25,] 0.0000000 0.6666667 0.3333333
## [26,] 0.0000000 1.0000000 0.0000000
## [27,] 0.0000000 0.3333333 0.6666667
## [28,] 0.0000000 0.3333333 0.6666667
## [29,] 0.0000000 0.6666667 0.3333333
## [30,] 0.0000000 0.0000000 1.0000000
```

```
predict_test_knn3 <- predict(knn_iris3, test_transformed, type='class')
confusionMatrix(predict_test_knn3, test_transformed$Species)
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction  setosa versicolor virginica
## setosa      10         0         0
## versicolor   0         7         5
## virginica    0         3         5
##
## Overall Statistics
##
##          Accuracy : 0.7333
```

```

##          95% CI : (0.5411, 0.8772)
##      No Information Rate : 0.3333
##      P-Value [Acc > NIR] : 8.752e-06
##
##          Kappa : 0.6
##
##      McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##          Class: setosa Class: versicolor Class: virginica
## Sensitivity          1.0000          0.7000          0.5000
## Specificity          1.0000          0.7500          0.8500
## Pos Pred Value       1.0000          0.5833          0.6250
## Neg Pred Value       1.0000          0.8333          0.7727
## Prevalence           0.3333          0.3333          0.3333
## Detection Rate       0.3333          0.2333          0.1667
## Detection Prevalence 0.3333          0.4000          0.2667
## Balanced Accuracy     1.0000          0.7250          0.6750

```

Using $k = 10$

```

# k=10
predict(knn_iris10, test_transformed, type='prob')

```

```

##      setosa versicolor virginica
## [1,] 1.0         0.0         0.0
## [2,] 1.0         0.0         0.0
## [3,] 1.0         0.0         0.0
## [4,] 1.0         0.0         0.0
## [5,] 1.0         0.0         0.0
## [6,] 1.0         0.0         0.0
## [7,] 1.0         0.0         0.0
## [8,] 1.0         0.0         0.0
## [9,] 1.0         0.0         0.0
## [10,] 1.0        0.0         0.0
## [11,] 0.0         0.2         0.8
## [12,] 0.0         0.7         0.3
## [13,] 0.0         0.6         0.4
## [14,] 0.0         0.5         0.5
## [15,] 0.0         0.5         0.5
## [16,] 0.0         0.4         0.6
## [17,] 0.0         0.8         0.2
## [18,] 0.1         0.7         0.2
## [19,] 0.0         0.6         0.4
## [20,] 0.0         0.7         0.3
## [21,] 0.0         0.5         0.5
## [22,] 0.0         0.5         0.5
## [23,] 0.0         0.3         0.7
## [24,] 0.0         0.5         0.5
## [25,] 0.0         0.3         0.7
## [26,] 0.0         0.8         0.2
## [27,] 0.0         0.6         0.4
## [28,] 0.0         0.6         0.4
## [29,] 0.0         0.4         0.6

```

```
## [30,]    0.0    0.4    0.6
predict_test_knn10 <- predict(knn_iris10, test_transformed, type='class')
confusionMatrix(predict_test_knn10, test_transformed$Species)

## Confusion Matrix and Statistics
##
##              Reference
## Prediction  setosa versicolor virginica
##   setosa      10          0          0
##   versicolor   0          8          5
##   virginica    0          2          5
##
## Overall Statistics
##
##              Accuracy : 0.7667
##              95% CI : (0.5772, 0.9007)
##   No Information Rate : 0.3333
##   P-Value [Acc > NIR] : 1.475e-06
##
##              Kappa : 0.65
##
##   McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##              Class: setosa Class: versicolor Class: virginica
## Sensitivity              1.0000              0.8000              0.5000
## Specificity              1.0000              0.7500              0.9000
## Pos Pred Value           1.0000              0.6154              0.7143
## Neg Pred Value           1.0000              0.8824              0.7826
## Prevalence               0.3333              0.3333              0.3333
## Detection Rate           0.3333              0.2667              0.1667
## Detection Prevalence     0.3333              0.4333              0.2333
## Balanced Accuracy        1.0000              0.7750              0.7000
```