STK4030
Statistical Learning: Advanced Regression and Classification

Compulsory Assignment

UiO : University of Oslo

Candidate Number: 33

November 22, 2015

Contents

Problem 1

The Wine Problem

(a) A statistical model in matrix form can be written as,

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{1.1}$$

The parameter **B** in equation-1.1 can estimated using OLS as,

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{1.2}$$

The statistical linear regression model in our case with quality as response and 11 different predictor variable can be written as,

$$\begin{aligned} \texttt{quality} = &\beta_0 + \beta_1\texttt{fixed.acidity} + \beta_2\texttt{volatile.acidity} + \beta_3\texttt{citric.acid} \\ &+ \beta_4\texttt{residual.sugar} + \beta_5\texttt{chlorides} + \beta_6\texttt{free.sulfur.dioxide} \\ &+ \beta_7\texttt{total.sulfur.dioxide} + \beta_8\texttt{density} + \beta_9\texttt{pH} \\ &+ \beta_9\texttt{pH} + \beta_10\texttt{sulphates} + \beta_11\texttt{alcohol} \end{aligned} \tag{1.3}$$

The model in equation - 1.3 is fitted using OLS, the summary output of the fitted model is,

```
lm.wine.formula <- makeFormula(X, y)
lm.wine <- lm(lm.wine.formula, data = wine[!test, ])
summary(lm.wine)



Call:
lm(formula = lm.wine.formula, data = wine[!test, ])

Residuals:
    Min      1Q  Median      3Q     Max
-3.2887 -0.5001 -0.0437  0.4053  2.1529

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)          75.202368  44.025271   1.708  0.08868 .
fixed.acidity         0.100442   0.088238   1.138  0.25594
volatile.acidity     -2.643420   0.574274  -4.603 6.25e-06 ***
citric.acid          -0.480493   0.449991  -1.068  0.28651
residual.sugar        0.068915   0.023401   2.945  0.00349 **
chlorides             4.402165   2.680560   1.642  0.10163
```

```
free.sulfur.dioxide    0.008849   0.003680   2.404  0.01684 *
total.sulfur.dioxide  -0.001924   0.001554  -1.238  0.21668
density              -75.363896  45.076996  -1.672  0.09563 .
pH                     0.410982   0.430460   0.955  0.34050
sulphates              0.406260   0.449993   0.903  0.36738
alcohol                0.349805   0.066077   5.294 2.38e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8291 on 288 degrees of freedom
Multiple R-squared:  0.2873,Adjusted R-squared:  0.2601
F-statistic: 10.56 on 11 and 288 DF,  p-value: 2.779e-16
```

Further, the model is used to predict the test observations and the mean square error for the prediction is found to be **0.586**. In addition, the summary output shows that **chlorides** has maximum positive (4.4) association while **density** has maximum negative (-75.36) association with the quality of the wine.

(b) The ridge estimate for equation - 1.1 is,

$$\hat{\beta}_{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \tag{1.4}$$

The shrinkage parameter λ in equation-1.4 is obtained through cross-validation procedure by minimizing the cross-validated test error. The model in eq-1.3 when fitted with ridge regression gives more shrinked coefficients. Although the coefficients are biased, they are closer to their true value. The model was fitted using `cv.glmnet` function from `glmnet` (Friedman et al., 2015) package with $\alpha = 0$ to get ridge estimate. The optimum tuning or shrinkage parameter λ for the ridge model is obtained from 10-fold cross-validation method.

```
rdg.wine <- cv.glmnet(X[!test, ], y[!test, ], alpha = 0, nfolds = 10)
rdg.wine.test.pred <- predict.cv.glmnet(object = rdg.wine, newx = X[test, ],
                                        s = 'lambda.min')
rdg.wine.train.pred <- predict.cv.glmnet(object = rdg.wine, newx = X[!test, ],
                                        s = 'lambda.min')
```

Here, prediction of test observations are made using lambda that has minimum test error. The mean squared test error is **0.563** and the coefficients for the ridge model are obtained as,

```
coef(rdg.wine)

12 x 1 sparse Matrix of class "dgCMatrix"
                                1
(Intercept)           5.949122884
fixed.acidity        -0.013659516
volatile.acidity     -0.059737056
citric.acid          -0.010767337
residual.sugar        0.016285074
chlorides            -0.035707368
free.sulfur.dioxide   0.032786213
```

```
total.sulfur.dioxide -0.038842851
density              -0.037222251
pH                    0.021696306
sulphates             0.009578018
alcohol               0.129656307
```

Here, the coefficient estimate for density has reduced (shrunk) considerably than linear model. The cross-validated mean squared error against the logarithm with base 10 of the tuning parameter (λ) values is presented in fig-1.1.

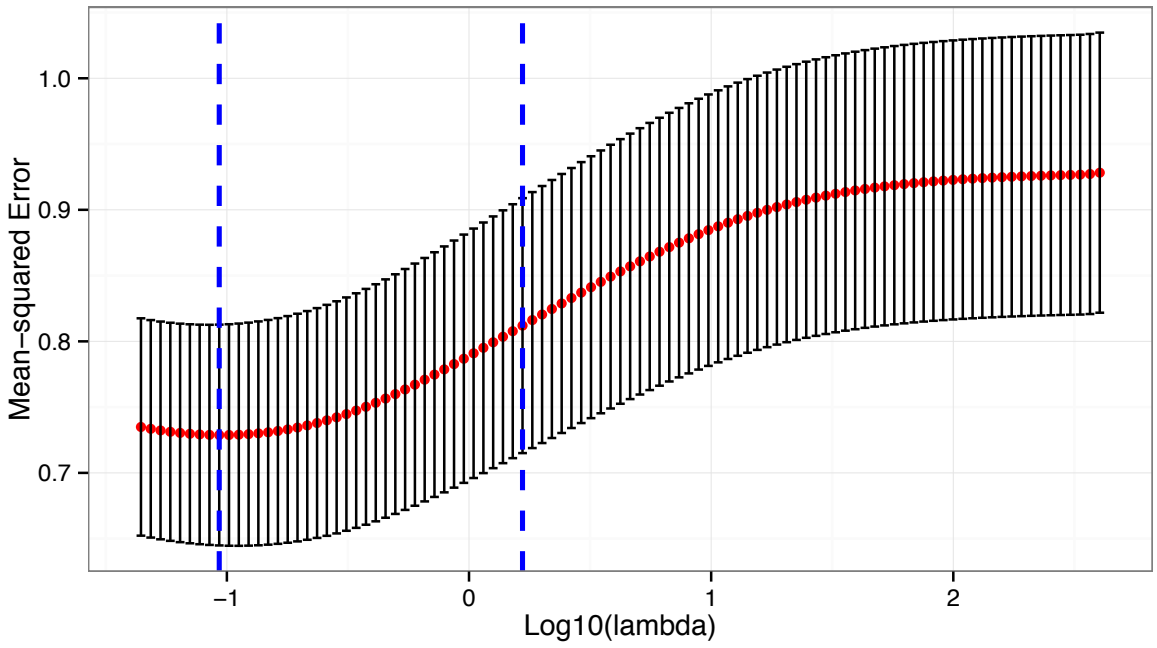


Figure 1.1: Cross-validated mean squared error against the logarithm with base 10 of the tuning parameter λ values for Ridge Regression Model

The optimum value of the tuning parameter (λ) is **0.093** with minimum mean cross-validated error (0.729).

(c) The same model in eq-1.3 is fitted using lasso model. The coefficient estimates for lasso regression can be written as (Hastie, Tibshirani, and Friedman, 2009),

$$\hat{\boldsymbol{\beta}}_{\text{lasso}} = \underset{\boldsymbol{\beta}}{\text{argmin}}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \ni \sum_{i=1}^{p} = |\boldsymbol{\beta}_j| \leq t \tag{1.5}$$

As lasso model not only shrinks the coefficients, but also can select variable setting their coefficients to zero, it also constitute variable selection properties like subset method. The model was fitted using `cv.glmnet` function from `glmnet` (Friedman et al., 2015) package with $\alpha = 1$. The optimum tuning or shrinkage parameter for lasso is obtained from 10-fold cross-validation method.

```
lso.wine <- cv.glmnet(X[!test, ], y[!test, ], alpha = 1, nfolds = 10)
lso.wine.test.pred <- predict.cv.glmnet(object = lso.wine, newx = X[test, ],
                                        s = 'lambda.min')
```

```
lso.wine.train.pred <- predict.cv.glmnet(object = lso.wine, newx = X[!test, ],
                                         s = 'lambda.min')
```

The prediction of test observation was made using lambda corresponding to minimum test error. The mean squared test error of **0.554** and the coefficients for the lasoo model are obtained as,

```
coef(lso.wine)

12 x 1 sparse Matrix of class "dgCMatrix"
                              1
(Intercept)           5.95571420
fixed.acidity          .
volatile.acidity     -0.06998582
citric.acid            .
residual.sugar         .
chlorides              .
free.sulfur.dioxide    .
total.sulfur.dioxide   .
density                .
pH                     .
sulphates              .
alcohol               0.27304749
```

Thus the coefficient estimates shows that only `volatile.acidity, alcohol` are sufficient for the prediction with error discussed above. The cross-validated mean squared error against the logarithm with base 10 of the tuning parameter (λ) values is presented in fig-1.2.

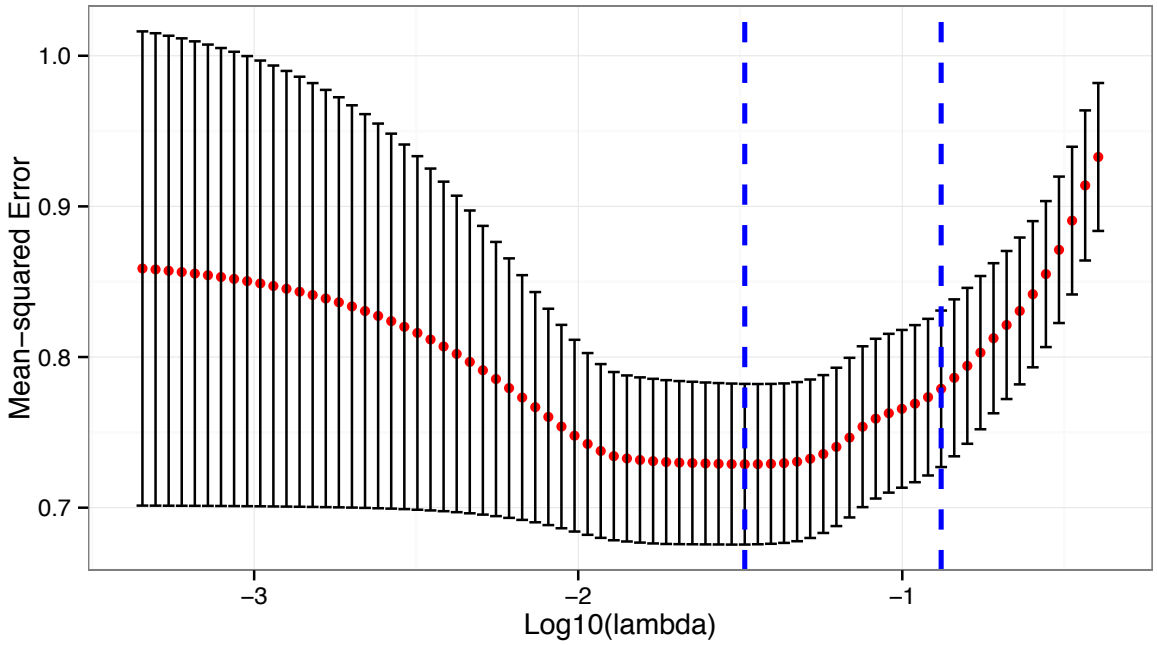


Figure 1.2: Cross-validated mean squared error against the logarithm with base 10 of the tuning parameter λ values for Lasso Regression Model

The optimum value of the tuning parameter (λ) is **0.033** with minimum mean cross-validated error (0.729).

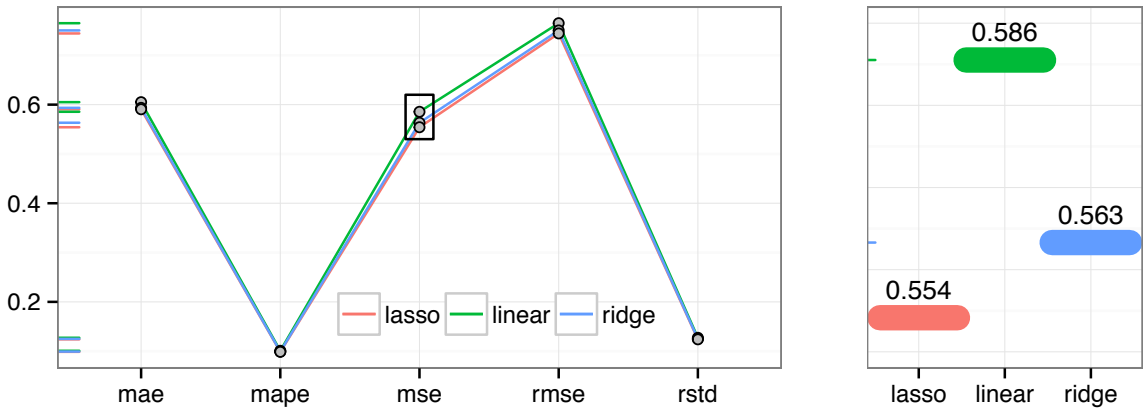


Figure 1.3: Comparison of Linear, Ridge and Lasso model with respect to various error of test prediction with MSE shown in right side

(d) The plot in figure-1.3 shows that Lasso regression model has the least error among others. The error present in the figure-1.3 are as follows which are obtained using `rmserr` function from `pracma` package of R.

`mae`	Mean Absolute Error	$\frac{1}{n}\sum\lvert y-\hat{y}\rvert$
`mse`	Mean Squared Error	$\frac{1}{n}\sum(y-\hat{y})^2$
`rmse`	Root Mean Squared Error	$\sqrt{\frac{1}{n}\sum(y-\hat{y})^2}$
`lmse`	Normalized Mean Squared Error	$\frac{1}{n}\sum\left\lvert\frac{y-\hat{y}}{y}\right\rvert$
`rstd`	relative Standard Deviation	$\frac{1}{\bar{y}}\sqrt{\frac{1}{n}\sum(y-\hat{y})^2}$

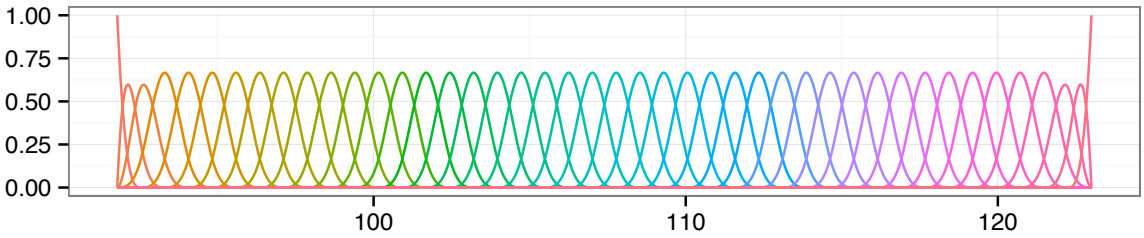
In the model the two covariates `volatile.acidity, alcohol` are selected, so, I would recommend the wine seler those two variables since they are highly responsible for the variation in the quality of different wines.

Problem 2

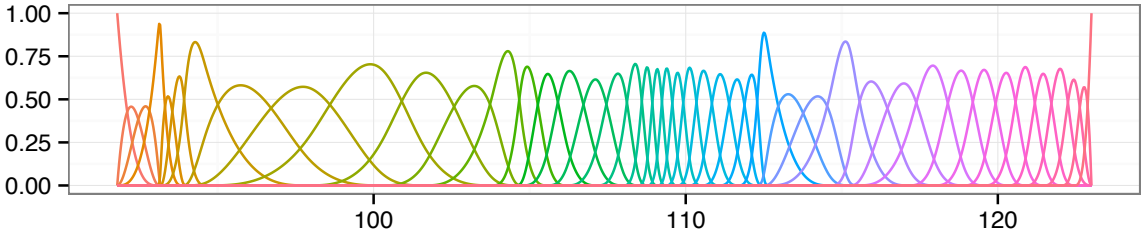
Strontium Ratio Problem

(a) The B-spline basis of order 4 (i.e. third degree) are constructed with external knots at range of age and 40 internal knots located (i) equidistantly, with 0.761 distance apart and (ii) at quantiles distribution of age. The plots in figure - 2.1a and figure - 2.1b, for these basis function are created for 1000 sample age sequenced between its range. The plots shows that the equidistant knots has constructed homogeneously distributed basis while the knots placed at quantiles has constructed more compact around dense data points.

```
plot(bs.eql); plot(bs.qtl)
```



(a) Knots equidistanced with 0.761 distance apart



(b) Knots located at the quantiles of distribution of age

Figure 2.1: B-splines for 1000 age samples sequenced between its range. These B-splines have 40 internal knots located and have boundry at the range of age

(b) A prediction is made using the basis obtained from knots placed at quantiles of age. A linear model fitted with strontium.ratio as response and the basis obtained as predictor. The smooth spline prediction is plotted from the two b-spline function considered above is shown in figure-2.2.

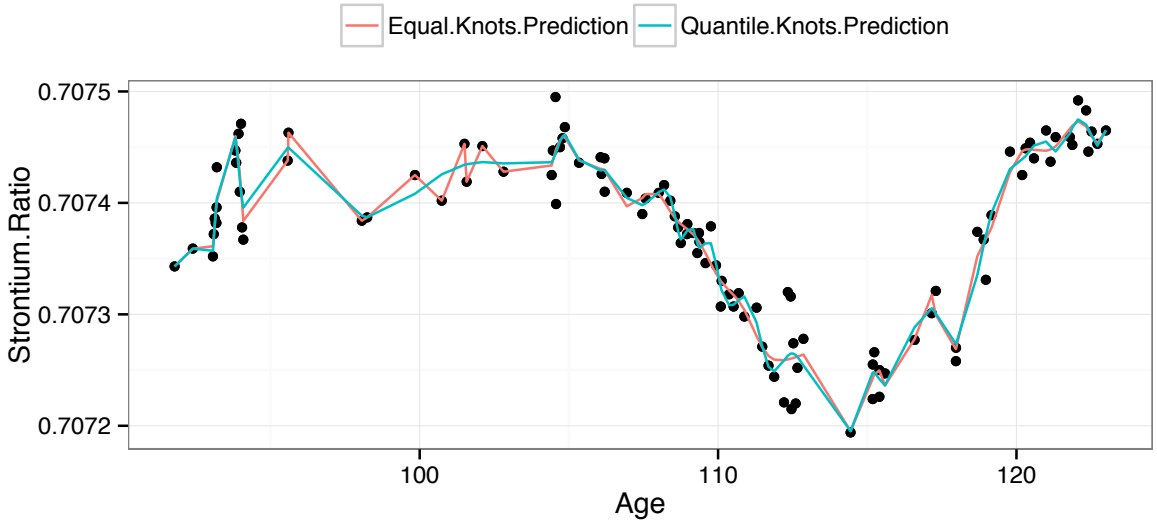


Figure 2.2: Prediction from the B-spline function with equally spaced knots and knots located at quantiles of the distribution of predictor variable age

Further, `smooth.spline` function from `stats` package from R is used to get the smooth spline where the smoothing parameter λ is selected using the leave-one-out cross-validation criteria and generalized cross-validation criteria (R Core Team, 2015). The fitted curve for both of these methods are plotted in figure-2.3.

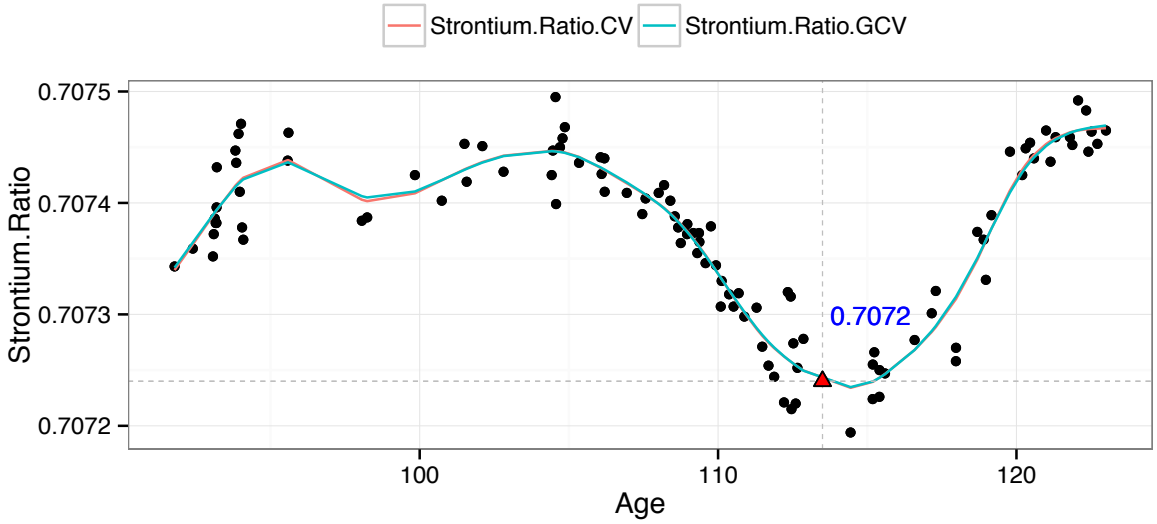


Figure 2.3: Spline Smooth curve fitted to the fossil data with the smoothing parameter obtained from leave-one-out cross-validation and Generalized cross-validation method

The smoothing parameter `spar` from leave-one-out cross-validation is obtained as 0.542 and that from GCV is 0.567. The coefficient λ of the integral of the squared second derivative in the fit (penalized log likelihood) criterion is a monotone function of these values (R Core Team, 2013) and is obtained to be 4.2×10^{-5} from leave-one-out cross-validation

and 6.3×10^{-5} from GCV.

The prediction for strontium ratio in a 113.5 million year old sample is obtained to be 0.71 using the model obtained from leave-one-out cross-validated model and almost same for the other one.

Problem 3

Faces Classification Problem

(a) The average of all portraits for each gender is plotted in figure-3.1. The average portraits differ for male and female. Although the average portraits has not given some discrete image of a person, one can easily differentiate the difference between male portrait and female portrait. In the average portraits each pixel is the average value intensity at the same pixel of all 100 portraits. The most dark and most light part such as hair and light background or highlighted parts can be easily distinguished.

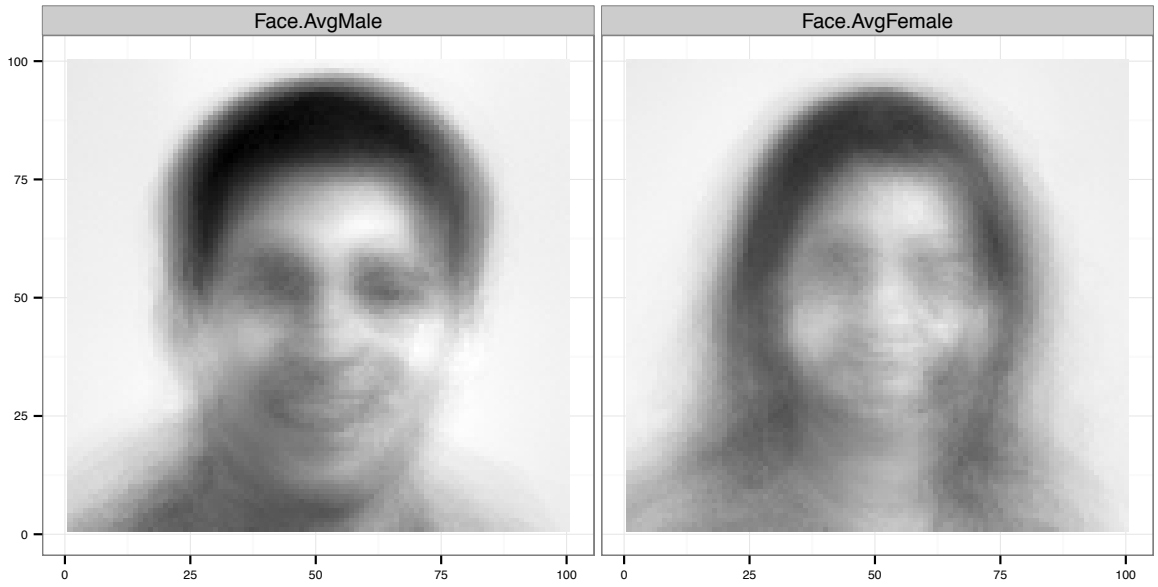


Figure 3.1: Average (Mean) of portraits for Male and Female

(b) From the `faces` matrix, principal component analysis (PCA) is performed. From the first three eigenvectors, eigen faces in fig-3.2 are obtained. In the eigen faces, the first one constitute most of the variation present in each pixel in all 200 images. The second one contains those variation which are left out by the first principle components which is always less than the first one. The similar is the case of third one. Here, each eigen faces are orthogonal to others.

```
pc.a <- prcomp(t(faces))
```

PCA shows that around 47 components are need to capture at least 80% of the variation present in the portraits.

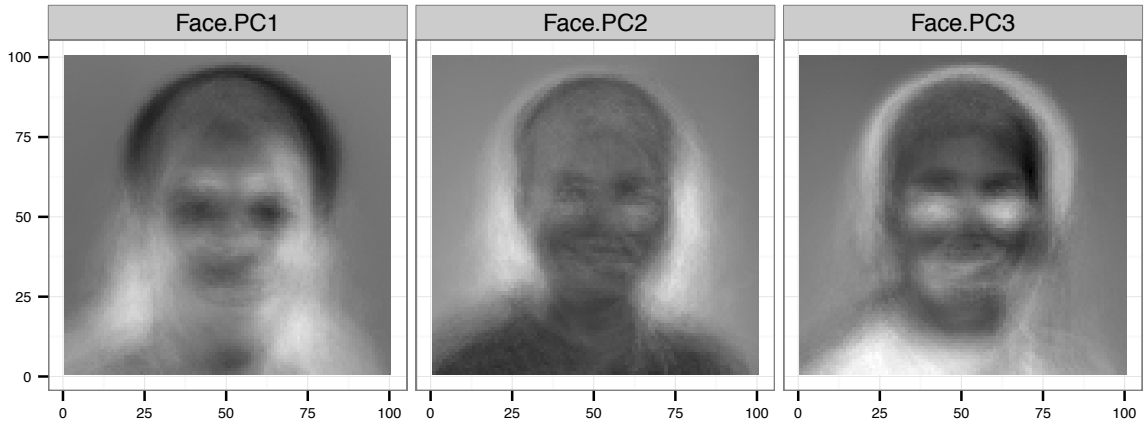


Figure 3.2: Eigen faces obtained from first three principal component for all 200 faces including both male and female

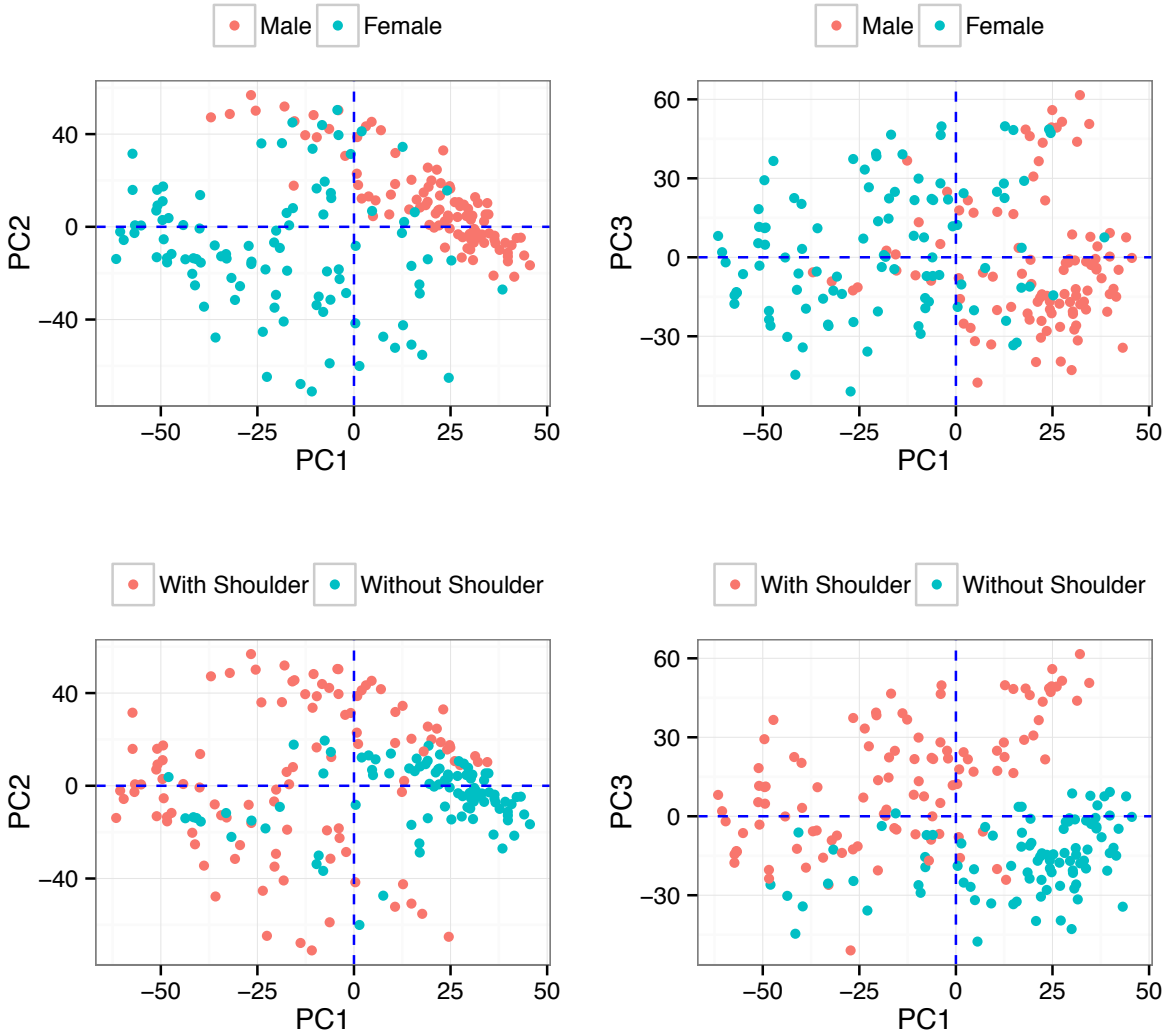


Figure 3.3: Principal components plots (Score plots) obtained from pca analysis colored according to Gender and Shoulder being present or not

(c) The plot in figure-3.3 clearly shows the distinction between male-female and person with shoulder and without shoulder. The plots in first row, i.e. PC1 against PC2 and PC3 shows that with some error, PC1 able to distinguish male and female. In the similar way PC1 also has classified person with shoulder and without shoulder with slightly more error than in the case of gender classification. These are only the first and second principle components which has captured 13.19%, 10.15% variation respectively presented in the portraits. With consideration of more components the classification can be better.

(d) Since the plot in figure-3.3 shows some classification of gender and shoulder being present or not, its desirable to perform some regression. The Gender variable is created as,

$$y_i = \begin{cases} -1, & \text{if } G_i = \text{male} \\ 1, & \text{if } G_i = \text{female} \end{cases} \tag{3.1}$$

A principal component regression (PCR) with y_i as responses is performed. The component is selected with leave-one-out cross-validation technique.

The number of principle components against root mean square error plotted in fig-3.4 shows that $m = 56$ components is needed to attain minimum error.

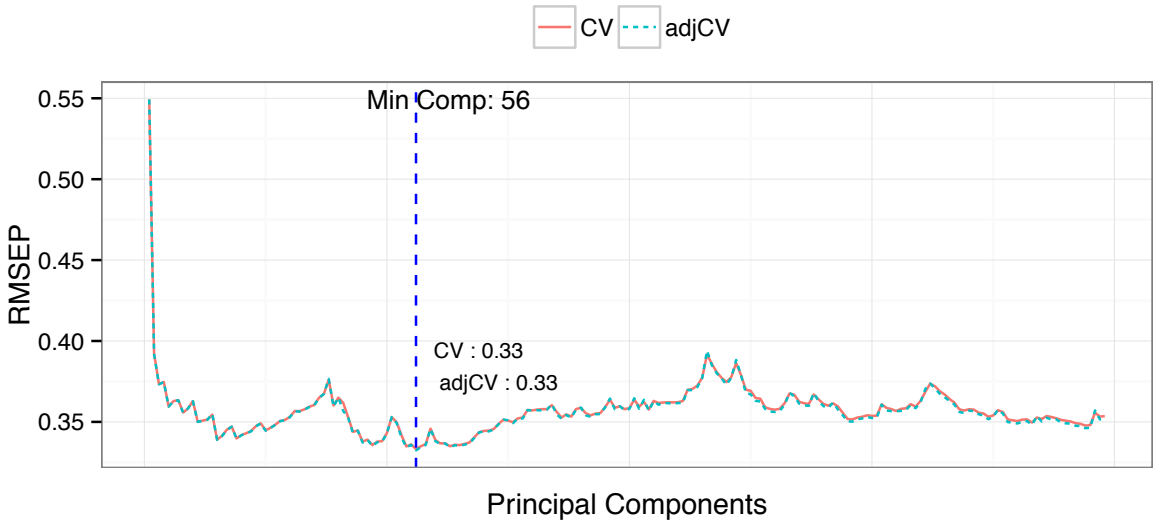


Figure 3.4: Number of principal component against Root Mean Square Error of Prediction plot for Principal Component Regression. The dashed line shows the number of component needed for minimum error

A prediction is made with 56 components and the following classification rule is applied.

$$\hat{y}_i = \begin{cases} \text{Female} & \text{if } \hat{f}(x_i) > 0 \\ \text{Male} & \text{if } \hat{f}(x_i) \leq 0 \end{cases} \tag{3.2}$$

A confusion matrix is created and plotted as in figure-3.5

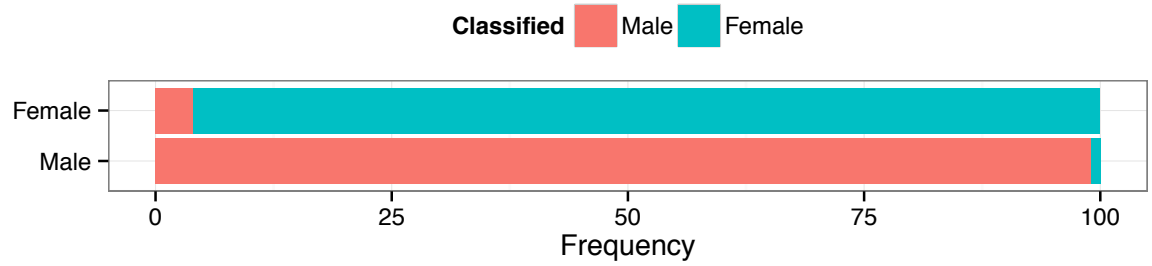


Figure 3.5: Confusion Plot for PCR classification

The model has given **0.025** as misclassification rate.

(e) The same classification is once more performed using partial least square (PLS) regression. With leave-one-out cross-validation methods is used to select the number of component required for minimum prediction error. The plot of number of principle component against root mean square error in fig-3.6 shows that $m = 2$ components is needed for minimum prediction error. The model has given **0.085** misclassification rate.

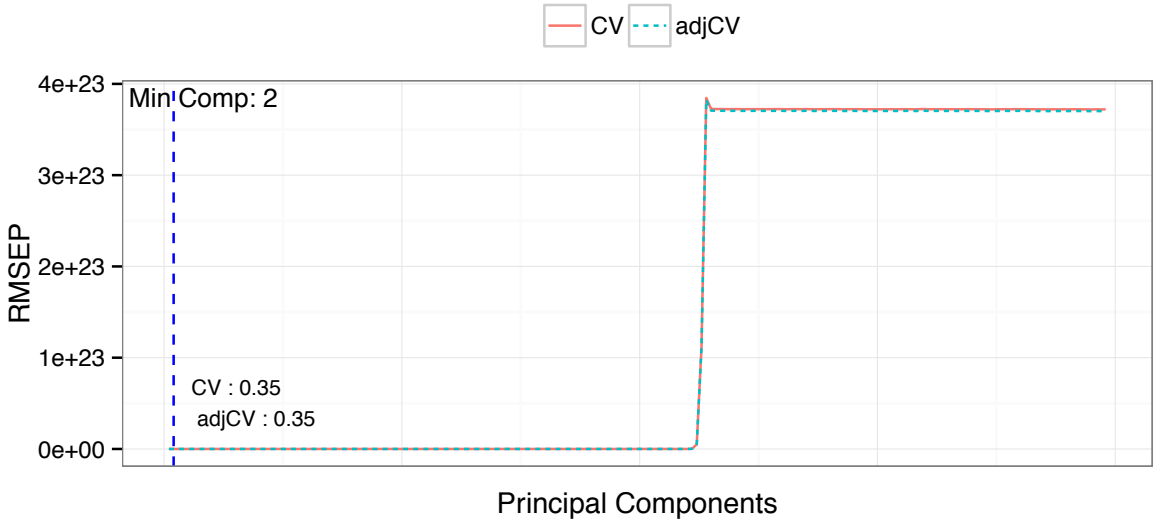


Figure 3.6: Number of principal component against Root Mean Square Error of Prediction plot for Partial Least Squre regression. The dashed line shows the number of component needed for minimum error

A prediction made with 2 components and applying classification rule in equation-3.2 gives a confusion matrix which is plotted in figure-3.7.

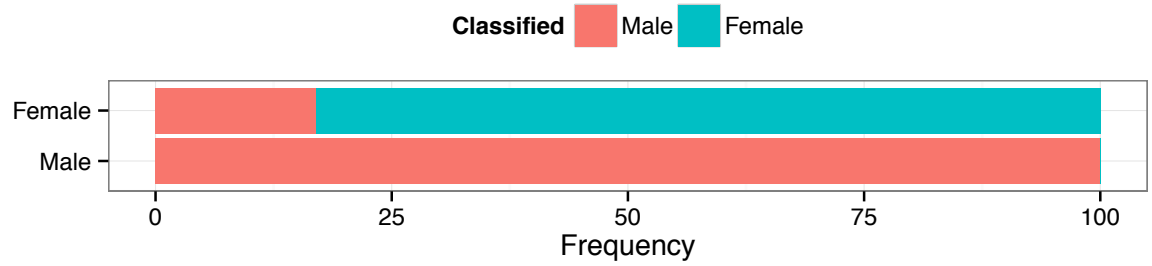


Figure 3.7: Confusion Plot for PLS classification

Here, the number of component needed for PLS is much less than that of PCR. Since, PCR accumulate the variation present on predictor variables on its initial principal components while PLS also tries to maximize the covariance between predictor and response. Since PLS being concentrate on classifying y rather than only capturing variation within response, it has used few principle components to do so in contrast to that of PCR.

(f) Since the principal components are the linear combination of all the variables, on considering five principal components we can capture 42.76% of variation present on faces dataset. These five principle component are taken into quadratic discriminant analysis (QDA) setup to create a classifier to classify gender as response variable.

Here the QDA model has classified the gender with 0.09 misclassification rate. The classification is shown in figure - 3.8 and figure - 3.9 with the decision boundary obtained from QDA. The figure present the same plot as in figure-3.3 but with the decision boundary obtained from the QDA analysis.

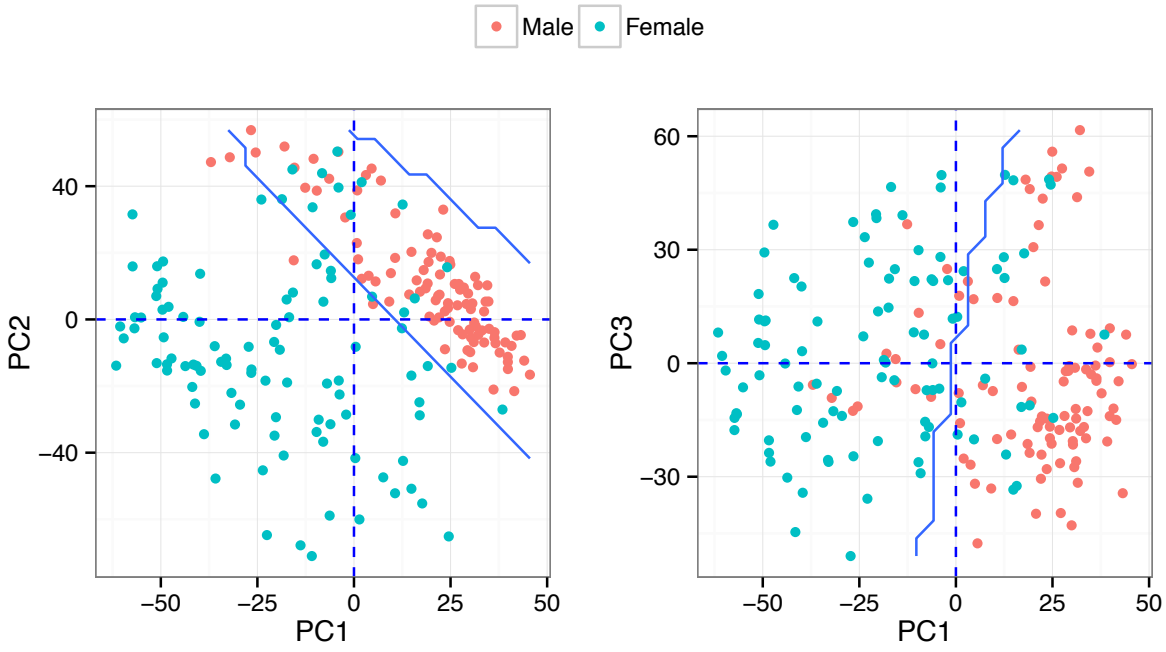


Figure 3.8: Classification of Gender from the `faces` data. The blue line is the decision boundry obtained from QDA.

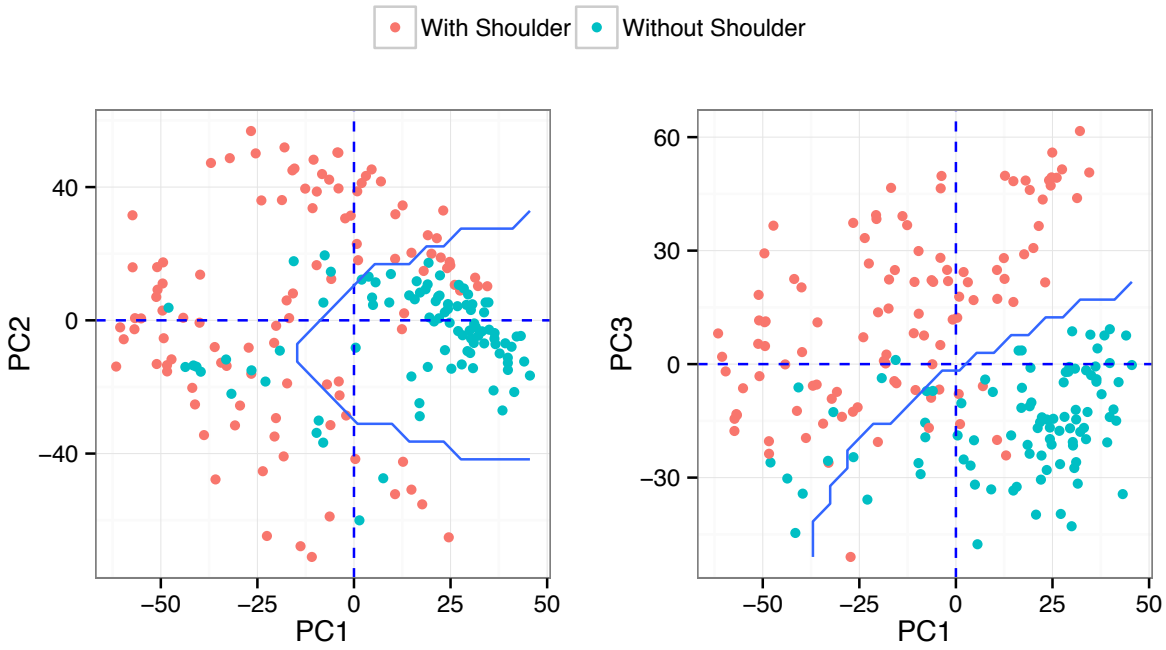


Figure 3.9: Classification of a person with shoulder and without shoulder from the `faces` data. The blue line is the decision boundry obtained from QDA.

(g) Further, both gender and shoulder categories are merged to create a category with 4 levels two for male with shoulder and without shoulder and same for female. The category is used as response for a new QDA setup with 5 principal components as predictors. The decision boundary is created for this situation where the boundaries has separated these 4 levels of response (figure - 3.10).

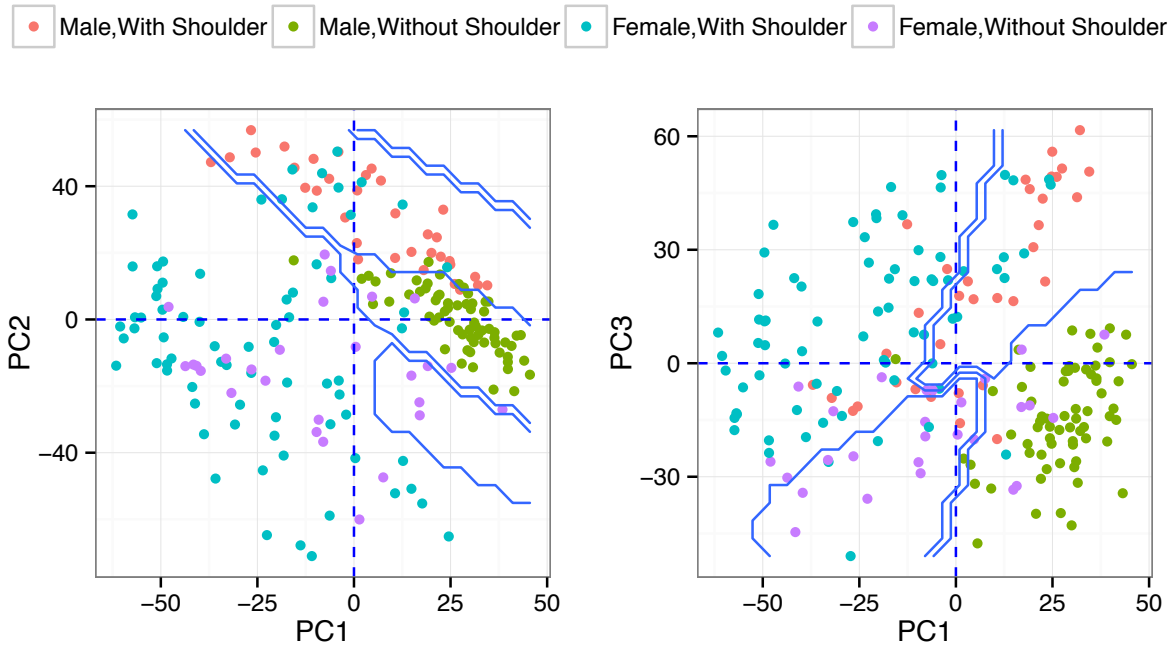


Figure 3.10: Decision Boundry for QDA on merged factor of gender and presence or absence of shoulder

Further the prediction obtained from this model is merged for the groups with shoulder and without shoulder within each gender. The classification obtained for gender in this situation gives 0.065 as misclassification rate. This is less than the misclassification made from considering only the gender variable. This can also be visualized from the score plot as in figure - 3.11 colored accordingly for this classification.

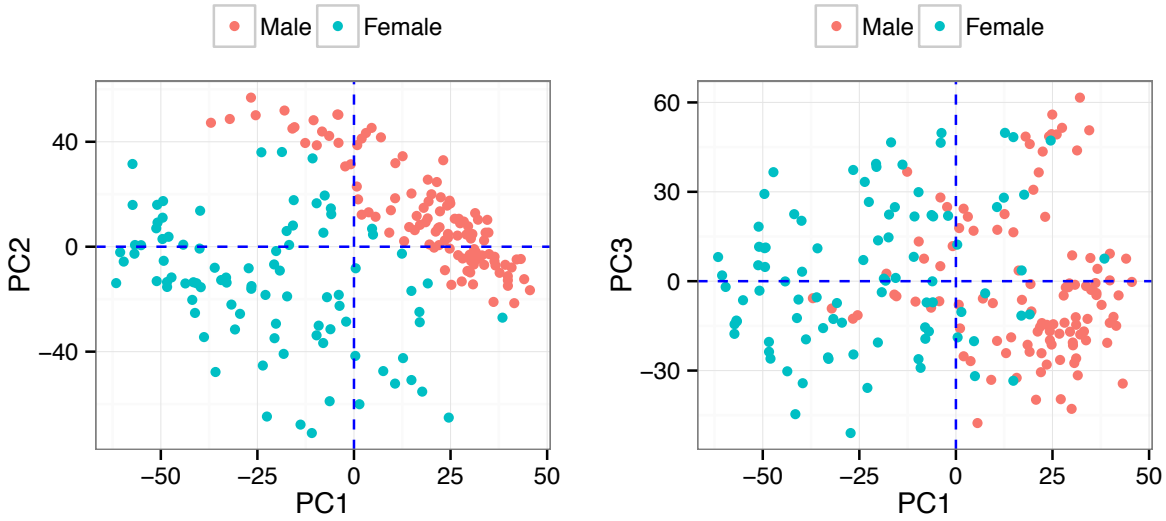


Figure 3.11: Decision Boundry for QDA on merged factor of persion having shoulder or not within gender variable as if it was classification for just gender.

(h) Regularized LDA is implemented on the full dataset with gender as response variable. Since the covariance matrix is regularized to solve the dimensional problem, regularized LDA can deal with large matrices where number of variables is larger than the observations. In a regularized LDA the covariance matrix is regularized as,

$$\mathbf{\Sigma} = (1 - \lambda)\mathbf{\Sigma} + \lambda\mathbf{I} \tag{3.3}$$

Due to high dimensional complexity, the full dataset could not be run in this setup however, only talking the 50 principal components, the `rda` function from `klaR` package (Roever et al., 2014) is used for regularized LDA. The model results in 0.01 as apparent error rate and 0.29 as cross-validated error rate which is much less than that obtained from un-regularized QDA. The decision boundary for this setup is plotted in figure - 3.12.

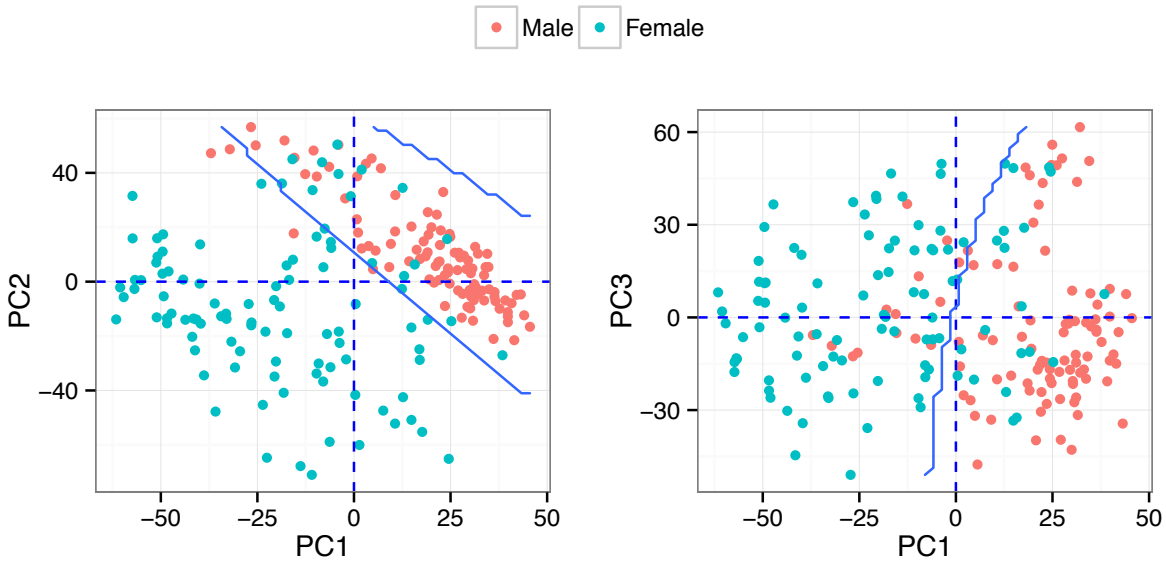


Figure 3.12: Classification of gender with decision boundry from Regularized Discreminant Analysis setup for scores of PC1 plotted against PC2 and PC3

Bibliography

[1] Baptiste Auguie. *gridExtra: Miscellaneous Functions for "Grid" Graphics*. R package version 2.0.0. 2015. URL: http://CRAN.R-project.org/package=gridExtra.

[2] Hans Werner Borchers. *pracma: Practical Numerical Math Functions*. R package version 1.8.6. 2015. URL: http://CRAN.R-project.org/package=pracma.

[3] M Dowle et al. *data.table: Extension of data.frame*. R package version 1.9.4. 2014. URL: http://CRAN.R-project.org/package=data.table.

[4] Jerome Friedman et al. *glmnet: Lasso and Elastic-Net Regularized Generalized Linear Models*. R package version 2.0-2. 2015. URL: http://CRAN.R-project.org/package=glmnet.

[5] Trevor J.. Hastie, Robert John Tibshirani, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009.

[6] Bjørn-Helge Mevik, Ron Wehrens, and Kristian Hovde Liland. *pls: Partial Least Squares and Principal Component Regression*. R package version 2.5-0. 2015. URL: http://CRAN.R-project.org/package=pls.

[7] R Core Team. *R: A Language and Environment for Statistical Computing*. ISBN 3-900051-07-0. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[8] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2015. URL: https://www.R-project.org/.

[9] Brian Ripley. *MASS: Support Functions and Datasets for Venables and Ripley's MASS*. R package version 7.3-43. 2015. URL: http://CRAN.R-project.org/package=MASS.

[10] Christian Roever et al. *klaR: Classification and visualization*. R package version 0.6-12. 2014. URL: http://CRAN.R-project.org/package=klaR.

[11] Hadley Wickham. *plyr: Tools for Splitting, Applying and Combining Data*. R package version 1.8.3. 2015. URL: http://CRAN.R-project.org/package=plyr.

[12] Hadley Wickham. *reshape2: Flexibly Reshape Data: A Reboot of the Reshape Package.* R package version 1.4.1. 2014. URL: http://CRAN.R-project.org/package=reshape2.

[13] Hadley Wickham and Winston Chang. *ggplot2: An Implementation of the Grammar of Graphics*. R package version 1.0.1. 2015. URL: http://CRAN.R-project.org/package=ggplot2.

[14] Yihui Xie. *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.11. 2015. URL: http://CRAN.R-project.org/package=knitr.

Appendix A

R-Codes

```
library(knitr)
opts_knit$set(root.dir = '..')
opts_chunk$set(comment = NA)
## Clear Everything
rm(list = ls())

## Load Sources and Packages
source('functions.R')
req.pkgs <- c('plyr', 'reshape2', 'ggplot2', 'glmnet', 'pls',
              'parallel', 'MASS', 'pracma', 'data.table', 'klaR',
              'splines', 'knitr', 'grid', 'gridExtra', 'base')
load.pkgs(req.pkgs)

## load and attach dataset
load('Data/wine.rdata')
load('Data/fossil.rdata')
load('Data/faces.rdata')
if (file.exists(file.path(getwd(), 'Exports', 'pcr.Rdata', fsep = '/')))
  load(file = 'Exports/pcr.Rdata')
if (file.exists(file.path(getwd(), 'Exports', 'pls.Rdata', fsep = '/')))
  load(file = 'Exports/pls.Rdata')
if (file.exists(file.path(getwd(), 'Exports', 'rda.Rdata', fsep = '/')))
  load(file = 'Exports/rda.Rdata')
## Scaling the data
y <- as.matrix(wine[, "quality"])
colnames(y) <- 'quality'
X <- as.matrix(wine[, 1:11], ncol = 11)
colnames(X) <- names(wine)[1:11]
X <- scale(X) ## Scaling the covariates
test <- wine$test ## Declare Test or Train observations
## Make Model Equation Part ------------
get.eq.part.1 <- paste(paste('\\beta', 1:3, sep = '_'),
                       paste('\\texttt{', names(wine)[1:3], '}', sep = ''),
                       collapse = '+')
get.eq.part.2 <- paste(paste('\\beta', 4:6, sep = '_'),
                       paste('\\texttt{', names(wine)[4:6], '}', sep = ''),
                       collapse = '+')
```

```
get.eq.part.3 <- paste(paste('\\beta', 7:9, sep = '_'),
                       paste('\\texttt{', names(wine)[7:9], '}', sep = ''),
                       collapse = '+')
get.eq.part.4 <- paste(paste('\\beta', 9:11, sep = '_'),
                       paste('\\texttt{', names(wine)[9:11], '}', sep = ''),
                       collapse = '+')
## -----------
lm.wine.formula <- makeFormula(X, y)
lm.wine <- lm(lm.wine.formula, data = wine[!test, ])
summary(lm.wine)
### Predict test sample
pred.wine.test <- predict(lm.wine, newdata = wine[test, ])
### Expected test error
wine.lm.test.err <- pracma::rmserr(pred.wine.test, y[test, ])
wine.lm.train.err <- pracma::rmserr(y[!test,], lm.wine$fitted.values)
### Covariate with strongest association
which.var.max.coef.p <- names(coef(lm.wine)[-1][which.max(coef(lm.wine)[-1])])
which.var.max.coef.n <- names(coef(lm.wine)[-1][which.min(coef(lm.wine)[-1])])
rdg.wine <- cv.glmnet(X[!test, ], y[!test, ], alpha = 0, nfolds = 10)
rdg.wine.test.pred <- predict.cv.glmnet(object = rdg.wine, newx = X[test, ],
                                        s = 'lambda.min')
rdg.wine.train.pred <- predict.cv.glmnet(object = rdg.wine, newx = X[!test, ],
                                        s = 'lambda.min')
wine.rdg.test.err <- rmserr(rdg.wine.test.pred, y[test, ])
wine.rdg.train.err <- rmserr(rdg.wine.train.pred, y[!test, ])
coef(rdg.wine)
### Preperation for plot
rdg.cv.wine.df <- data.frame(
  log.lmda = log10(rdg.wine$lambda),
  MSE = rdg.wine$cvm,
  upper = rdg.wine$cvup,
  lower = rdg.wine$cvlo
)
rdg.cv.range.df <- data.frame(
  lo = log10(rdg.wine$lambda.min),
  hi = log10(rdg.wine$lambda.1se)
)

### The plot log10(lambda) vs MSE
ridge.mse.lambda.plot <- ggplot(rdg.cv.wine.df, aes(log.lmda, MSE)) +
  geom_point(color = 'red') +
  geom_errorbar(aes(ymax = upper, ymin = lower)) +
  theme_bw() +
  labs(x = 'Log10(lambda)', y = 'Mean-squared Error') +
  geom_vline(data = rdg.cv.range.df, aes(xintercept = c(lo, hi)),
             color = 'blue',
             linetype = 2, size = 1)
print(ridge.mse.lambda.plot)
lso.wine <- cv.glmnet(X[!test, ], y[!test, ], alpha = 1, nfolds = 10)
lso.wine.test.pred <- predict.cv.glmnet(object = lso.wine, newx = X[test, ],
                                        s = 'lambda.min')
```

```
lso.wine.train.pred <- predict.cv.glmnet(object = lso.wine, newx = X[!test, ],
                                         s = 'lambda.min')
wine.lso.test.err <- rmserr(lso.wine.test.pred, y[test, ])
wine.lso.train.err <- rmserr(lso.wine.train.pred, y[!test, ])
coef(lso.wine)
### Preperation for plot
lso.cv.wine.df <- data.frame(
  log.lmda = log10(lso.wine$lambda),
  MSE = lso.wine$cvm,
  upper = lso.wine$cvup,
  lower = lso.wine$cvlo
)
lso.cv.range.df <- data.frame(
  lo = log10(lso.wine$lambda.min),
  hi = log10(lso.wine$lambda.1se)
)

### The plot log10(lambda) vs MSE
lso.mse.lambda.plot <- ggplot(lso.cv.wine.df, aes(log.lmda, MSE)) +
  geom_point(color = 'red') +
  geom_errorbar(aes(ymax = upper, ymin = lower)) +
  theme_bw() +
  labs(x = 'Log10(lambda)', y = 'Mean-squared Error') +
  geom_vline(data = lso.cv.range.df, aes(xintercept = c(lo, hi)),
             color = 'blue',
             linetype = 2, size = 1)
print(lso.mse.lambda.plot)
test.err.dt <- data.table(melt(
  list(linear = wine.lm.test.err,
       ridge = wine.rdg.test.err,
       lasso = wine.lso.test.err)
))
setnames(test.err.dt, names(test.err.dt), c('TestError', 'ErrorType', 'Model'))
setkeyv(test.err.dt, c('Model', 'ErrorType'))
train.err.dt <- data.table(melt(
  list(linear = wine.lm.train.err,
       ridge = wine.rdg.train.err,
       lasso = wine.lso.train.err)
))
setnames(train.err.dt, names(train.err.dt), c('TrainError', 'ErrorType', 'Model'))
setkeyv(train.err.dt, c('Model', 'ErrorType'))

err.dt <- melt(test.err.dt[train.err.dt], 2:3,
               variable.name = 'Which.Error',
               value.name = 'Error')

mdl.comp <- ggplot(dplyr::filter(melt(list(linear = wine.lm.test.err,
                                           ridge = wine.rdg.test.err,
                                           lasso = wine.lso.test.err)),
                                 L2 != 'nmse'),
              aes(L2, value)) +
```

```
  geom_line(aes(group = L1, color = L1)) +
  geom_point(shape = 21, fill = 'gray') +
  theme_bw() +
  theme(legend.position = c(0.6, 0.15),
        legend.title = element_blank(),
        legend.direction = 'horizontal',
        axis.title = element_blank(),
        legend.background = element_blank()) +
  geom_rug(side = 'l', aes(color = L1)) +
    annotate(geom = 'rect', xmin = 2.9, xmax = 3.1,
             ymin = 0.53, ymax = 0.62,
             alpha = 0, color = 'black')
msedf <- test.err.dt[ErrorType == 'mse']
mdl.comp.mse <- ggplot(msedf, aes(Model, TestError, color = Model)) +
    geom_violin(size = 5) + theme_bw() +
    theme(legend.position = 'none',
          axis.title = element_blank(),
          axis.text.y = element_blank(),
          axis.ticks.y = element_blank()) +
    scale_y_continuous(limits = c(0.55, 0.59)) +
    labs(y = 'Mean Squared Error (MSE)') +
    geom_rug(sides = 'l') +
    geom_text(aes(label = round(TestError, 3)),
              vjust = -1, color = 'black', size = 4)
gridExtra::grid.arrange(mdl.comp, mdl.comp.mse, ncol = 2, widths = c(7/10, 3/10))
fossil <- data.table(fossil)
n.knots <- 40
n.order <- 4
## On generating 40 knots, the distance between them equals to 0.7613 âĽĹ 0.761
equal.knots <- seq(fossil[, min(age)],
                   fossil[, max(age)],
                   length.out = 42)[2:41]
## Quantile spaced knots is generated by bs function on specifying df
bs.eql <- bs(x = fossil[, age],  df = n.knots + n.order + 1,
             knots = equal.knots,  degree = n.order - 1, intercept = TRUE)
bs.qtl <- bs(x = fossil[, age], df = n.knots + n.order,
             degree = n.order - 1, intercept = TRUE)

## Fitting Model
eql.knot.model <- lm(strontium.ratio ~ bs.eql, data = fossil)
qtl.knot.model <- lm(strontium.ratio ~ bs.qtl, data = fossil)

pred.dt <- melt(
  data.table(Age = fossil[, age],
             Strontium.Ratio = fossil[, strontium.ratio],
             Equal.Knots.Prediction = eql.knot.model$fitted.values,
             Quantile.Knots.Prediction = qtl.knot.model$fitted.values),
  id.vars = 1:2)

predplot <- ggplot(pred.dt, aes(Age, Strontium.Ratio)) +
  geom_point() +
```

```
  geom_line(aes(color = variable, y = value)) +
  theme_bw() +
  theme(legend.title = element_blank(),
        legend.position = 'top')
plot(bs.eql); plot(bs.qtl)
print(predplot)
spline.cv <- smooth.spline(x = fossil[, age],
                           y = fossil[, strontium.ratio],
                           cv = TRUE,
                           nknots = 40,
                           all.knots = FALSE)
spline.gcv <- smooth.spline(x = fossil[, age],
                           y = fossil[, strontium.ratio],
                           cv = FALSE,
                           nknots = 40,
                           all.knots = FALSE)
spline.fit <- data.table(Age = fossil[, age],
                         Strontium.Ratio = fossil[, strontium.ratio],
                         spline.fit.age = spline.cv$x,
                         Strontium.Ratio.CV = spline.cv$y,
                         Strontium.Ratio.GCV = spline.gcv$y)
newx <- 113.5
newYpred <- data.frame(x = newx,
                       y = c(predict(spline.cv, x = newx)$y,
                             predict(spline.gcv, x = newx)$y))


splineplot <- ggplot(melt(spline.fit, 1:3), aes(Age, Strontium.Ratio)) +
  geom_point() +
  geom_line(aes(x = spline.fit.age,
                y = value, color = variable)) +
  theme_bw() + theme(legend.title = element_blank(), legend.position = 'top') +
  geom_hline(yintercept = newYpred$y, col = 'gray', linetype = 2, size = 0.25) +
  geom_vline(xintercept = newYpred$x, col = 'gray', linetype = 2, size = 0.25) +
  annotate(geom = 'text', newYpred$x, newYpred$y,
           label = round(newYpred$y, 4),
           vjust = -3, hjust = -0.1, size = 4, color = 'blue', bg = 'white') +
  geom_point(data = newYpred, aes(x = x, y = y),
             fill = 'red', size = 2.5, shape = 24)
print(splineplot)
## Using DataTable
faces.dt <- data.table(faces)
## Create Sex Logical Variable ------------------------------------------------
Male <- rep(c(TRUE, FALSE), each = 100)
shoulder <- as.logical(shoulder)
attr.dt <- data.table(
  Gender = factor(Male,
                  levels = c('TRUE', 'FALSE'),
                  labels = c('Male', 'Female')),
  Shoulder = factor(shoulder,
                  levels = c('TRUE', 'FALSE'),
```

```
                  labels = c('With Shoulder',
                             'Without Shoulder'))
  )
## Factor Combining to create new one
invisible(attr.dt[, Gender.Shoulder := dae::fac.combine(list(
  attr.dt[, Gender], attr.dt[, Shoulder]), combine.levels = T)
  ])
## Plot average faces
avgFaces <- faces.dt[, list(AvgMale = rowMeans(.SD[, Male, with = F]),
                            AvgFemale = rowMeans(.SD[, !Male, with = F]))]
getFaced(as.matrix(avgFaces), n.faces = 1:2, facet.ncol = 2)
pc.a <- prcomp(t(faces))
pc.score <- data.table(pc.a$x)
pc.rotate <- data.table(pc.a$rotation)
getFaced(as.matrix(pc.rotate), n.faces = 1:3, facet.ncol = 3)
listGrid <- expand.grid(list(c(1,2), c(1,3)), c('Gender', 'Shoulder'))
score.plt <- mlply(listGrid, function(Var1, Var2){
  Var1 <- unlist(Var1); Var2 <- as.character(Var2)
  getScored(pc.score, ncomp = 1:3,
            which = Var1,
            attr.df = attr.dt,
            col.var = Var2)
})
getGrided(score.plt)
## Creating Gender Response -------
Gender <- ifelse(Male, -1, 1)
Shoulder <- ifelse(shoulder, -1, 1)
Gender.Shoulder <- attr.dt[, as.numeric(Gender.Shoulder)]
if (!('pc.r' %in% ls()) | !('pls.r' %in% ls()))
  pls.options(parallel = makeCluster(6, type = "PSOCK"))
if (!('pc.r' %in% ls())) {
  pc.r <- pcr(Gender ~ t(faces), validation = 'LOO')
  save(pc.r, file = 'Exports/pcr.Rdata')
}
if (!('pls.r') %in% ls()) {
  pls.r <- plsr(Gender ~ t(faces), validation = 'LOO')
  save(pls.r, file = 'Exports/pls.Rdata')
}
if (!('pc.r' %in% ls()) | !('pls.r' %in% ls()))
  stopCluster(pls.options()$parallel)
pcr.msep <- adply(MSEP(pc.r)$val, 3)[-1, ]
pcr.msep[, 1] <- as.numeric(pcr.msep[, 1]) - 1

pcr.min.comp <- which.min(pcr.msep$adjCV)
plotRMSEP(MSEP(pc.r))
getClassified(pc.r$fitted.values[, , pcr.min.comp], Gender)$conf.plot
pls.msep <- adply(MSEP(pls.r)$val, 3)[-1, ]
pls.msep[, 1] <- as.numeric(pls.msep[, 1]) - 1

pls.min.comp <- which.min(pls.msep$adjCV)
plotRMSEP(MSEP(pls.r))
```

```
getClassified(pls.r$fitted.values[, , pls.min.comp], Gender)$conf.plot
## Quadratic Decision Analysis
pc.model.mat <- data.frame(pc.score, attr.dt)
qda.md.name <- c('Gender', 'Shoulder', 'Gender.Shoulder')
qda.fit <- llply(qda.md.name, function(x){
  qda(resp ~ ., data = data.frame(resp = eval(parse(text = x)),
                                  pc.score[, 1:5, with = F]))
})
names(qda.fit) <- qda.md.name
qda.score.plot <- mlply(listGrid, function(Var1, Var2) {
    Var1 <- unlist(Var1); Var2 <- as.character(Var2)
    if (Var2 == 'Gender')
      qdb <- getDB(pc.score, grid.size = 25,
                   da.fit = qda.fit$Gender, n.comp = Var1)
    if (Var2 == 'Shoulder')
      qdb <- getDB(pc.score, grid.size = 25,
                   da.fit = qda.fit$Shoulder, n.comp = Var1)
    plt <- getScored(pc.score, ncomp = 1:3,
                     which = Var1,
                     attr.df = attr.dt,
                     col.var = Var2)
    plt <- plt + geom_contour(data = qdb,
                              aes_string(paste('PC', Var1, sep = ''), z = 'z'),
                              bins = 1)
    return(plt)
  })
cf.df.qda.gender <- table(predict(qda.fit$Gender)$class, Gender)
msc.rate.qda <- 1 - sum(diag(cf.df.qda.gender))/sum(cf.df.qda.gender)
qda.score.plot$ncol <- 2
do.call(grid_arrange_shared_legend, qda.score.plot[c(1:2, 5)])
do.call(grid_arrange_shared_legend, qda.score.plot[3:5])
## Question 3(g) ------------------------
qda.gs.plot <- llply(list(c(1,2), c(1,3)), function(x){
  GS.db <- getDB(pc.score, grid.size = 25, da.fit = qda.fit$Gender.Shoulder,
                 n.comp = x)
  plt <- getScored(scores(pc.a), ncomp = 1:3,
                   which = x,
                   attr.df = attr.dt,
                   col.var = 'Gender.Shoulder')
  plt <- plt + geom_contour(data = GS.db, aes_string(names(GS.db)[1:2], z = 'z'),
                            lineend = 'round', linejoin = 'round', linetype = 1,
                            bins = 3)
  return(plt)
})
qda.gs.plot$ncol <- 2

## Classifications
qda.gs.hat <- predict(qda.fit$Gender.Shoulder)$class
qda.gs.hat <- factor(qda.gs.hat, levels = 1:4,
                     labels = levels(attr.dt$Gender.Shoulder))
cf.gs.tbl <- table(attr.dt$Gender.Shoulder, qda.gs.hat)
```

```
error.rate <- 1 - sum(diag(cf.gs.tbl)) / sum(cf.gs.tbl)

attr.plus <- data.frame(attr.dt, Gender.Shoulder.Fitted = qda.gs.hat)

## Merging within Gender
qda.gender.fit2 <- factor(ifelse(grepl('Male', qda.gs.hat), 'Male', 'Female'),
                          levels = c('Male', 'Female'))
cf.qda.gender.tbl <- table(qda.gender.fit2, attr.dt$Gender)
msc.rate.qda2 <- 1 - sum(diag(cf.qda.gender.tbl)) / sum(cf.qda.gender.tbl)
do.call(grid_arrange_shared_legend, qda.gs.plot)
qda.gs.plot2 <- llply(list(c(1,2), c(1,3)), function(x){
    plt <- getScored(scores(pc.a), ncomp = 1:3,
                     which = x,
                     attr.df = data.table(Gender = qda.gender.fit2),
                     col.var = 'Gender')
    return(plt)
})
qda.gs.plot2$ncol <- 2
do.call(grid.arrange, qda.gs.plot2)
if (!('rda.fit' %in% ls())) {
  rda.fit <- rda(resp ~ .,
                 data = data.frame(resp = Gender,
                                   pc.score[, 1:50, with = F]),
                 regularization = c(lambda = 0, gamma = 'gamma'),
                 crossval = TRUE,
                 fold = nrow(pc.score))
  save(rda.fit, file = 'Exports/rda.Rdata')
}
rda.gs.plot <- llply(list(c(1,2), c(1,3)), function(x){
    GS.db <- getDB(pc.score, grid.size = 50, da.fit = rda.fit,
                   n.comp = x)
    plt <- getScored(scores(pc.a), ncomp = 1:3,
                     which = x,
                     attr.df = attr.dt,
                     col.var = 'Gender')
    plt <- plt + geom_contour(data = GS.db, aes_string(names(GS.db)[1:2], z = 'z'),
                              lineend = 'round', linejoin = 'round', linetype = 1,
                              bins = 1)
    return(plt)
})
rda.gs.plot$ncol <- 2
do.call(grid_arrange_shared_legend, rda.gs.plot)
```

Appendix B

R-functions

```
load.pkgs <- function(pkgs.list){
  req.pkgs <- unlist(pkgs.list)
  invisible(lapply(req.pkgs, require,
                   quietly = TRUE,
                   warn.conflicts = FALSE,
                   character.only = TRUE))
}

makeFormula <- function(X, y){
  y <- data.frame(y)
  X <- data.frame(X)
  as.formula(paste(names(y), paste(names(X), collapse = '+'), sep = '~'))
}

getFaced <- function(faceMat, n.faces = as.matrix(1:1),
                     facet.ncol = floor(sqrt(length(n.faces)))) {
  if (is.vector(faceMat)) {
    faceMat <- as.matrix(faceMat)
  }
  face.dim <- sqrt(nrow(faceMat))
  faceGrid <- expand.grid(x = face.dim:1, y = face.dim:1)
  faceGrid <- data.frame(faceGrid, Face = faceMat[, n.faces])
  faceStack <- melt(faceGrid, c('x', 'y'))

  plt <- ggplot(faceStack, aes_string('y', 'x', fill = 'value')) +
    geom_tile() +
    facet_wrap(~variable, ncol = facet.ncol) +
    scale_fill_continuous(low = 'black', high = 'white') +
    theme_bw() + theme(axis.title = element_blank(),
                       axis.text = element_text(size = 6),
                       legend.position = 'none')
  print(plt)
}

getScored <- function(scoreMat, ncomp, which = c(1,2), attr.df = NULL,
                      col.var = NULL, shape.var = NULL){
  require(data.table)
```

```
  if (!is.null(attr.df)) {
    scores.dt <- data.table(scoreMat[,], attr.df)
  } else {
    scores.dt <- data.table(scoreMat[,])
  }
  pc.n <- names(scores.dt[, which, with = F])
  ggplot(scores.dt, aes_string(pc.n[1], pc.n[2])) +
    geom_point(aes_string(color = col.var, shape = shape.var)) +
    theme_bw() +
    theme(legend.title = element_blank(),
          legend.position = 'top') +
    geom_vline(xintercept = 0, col = 'blue', linetype = 2) +
    geom_hline(yintercept = 0, col = 'blue', linetype = 2)
}

getGrided <- function(gp.lst, ncol = floor(sqrt(length(gp.lst)))){
  gp.lst$ncol <- ncol
  do.call(gridExtra::grid.arrange, gp.lst)
}

getClassified <- function(fitted.value, original.value,
                          rule = function(x){ifelse(x < 0, -1, 1)}){
  classified.value = rule(fitted.value)
  confusion.matrix = table(classified.value, original.value)
  errors <- confusion.matrix[row(confusion.matrix) != col(confusion.matrix)]
  error.rate = sum(errors) / sum(confusion.matrix)
  dimnames(confusion.matrix) <- list(Classified = c('Male', 'Female'),
                                     Original = c('Male', 'Female'))

  class.plt <- ggplot(as.data.frame(confusion.matrix),
                      aes(Original, y = Freq, fill = Classified)) +
    geom_bar(stat = 'identity', size = 0.5) +
    theme_bw() + theme(legend.position = 'top',
                       axis.title.y = element_blank()) +
    coord_flip() +
    labs(y = 'Frequency')

  return(invisible(list(classified.value = classified.value,
                        confusion.matrix = confusion.matrix,
                        conf.plot = class.plt,
                        error.rate = error.rate)))
}

plotRMSEP <- function(rmsep.obj, h.just = -.2){
  df <- adply(rmsep.obj$val, 3)[-1, ]
  df[, 1] <- as.numeric(df[, 1]) - 1
  names(df) <- c('Comp', 'CV', 'adjCV')

  df.stk <- melt(df, 1, variable.name = 'CV', value.name = 'RMSEP')
  df.stk.min <- ddply(df.stk, .(CV), dplyr::filter, RMSEP == min(RMSEP))
```

```
  ### Generating Plot
  plt <- ggplot(df.stk, aes(Comp, RMSEP, group = CV, color = CV)) +
    geom_line(aes(linetype = CV)) + theme_bw() +
    theme(axis.text.x = element_blank(),
          axis.ticks.x = element_blank(),
          legend.position = 'top',
          legend.title = element_blank()) +
    labs(x = 'Principal Components') +
    geom_vline(data = df.stk.min, aes(xintercept = Comp),
               color = 'blue', linetype = 2) +
    annotate(geom = 'text', x = df.stk.min$Comp,
             y = unlist(daply(df.stk, .(CV), summarize, median(RMSEP))),
             label = paste(df.stk.min$CV, ':',
                           round(df.stk.min$RMSEP, 2)),
             hjust = h.just, vjust = c(-3.5,-1.5),
             size = 3) +
    annotate('text', mean(df.stk.min$Comp), max(df.stk$RMSEP),
             label = paste('Min Comp:', df.stk.min$Comp[1]),
             size = 4, hjust = h.just + 0.5)
  return(plt)
}

getDB <- function(score.df, grid.size, da.fit, n.comp){
  gs <- grid.size
  da.fit <- update(da.fit, data = model.frame(da.fit)[, c(1, n.comp + 1)])
  mmpc <- ldply(c(min, max), function(x)
    apply(score.df[, n.comp, with = F], 2, x))
  rownames(mmpc) <- c('min', 'max')

  gsamp <- llply(seq_along(n.comp), function(x){
    seq(mmpc["min", names(score.df[, n.comp, with = F])[x]],
        mmpc["max", names(score.df[, n.comp, with = F])[x]],
        length.out = gs)
  })
  names(gsamp) <- names(score.df[, n.comp, with = F])
  gdf <- do.call(expand.grid, gsamp)
  yhat <- as.numeric(as.character(predict(da.fit, gdf)$class))
  qda.db <- data.frame(gdf, z = yhat)
  return(qda.db)
}

## From Hadley Wickham (github)
grid_arrange_shared_legend <- function(..., ncol = 1) {
  plots <- list(...)
  g <- ggplotGrob(plots[[1]] + theme(legend.position = "top"))$grobs
  legend <- g[[which(sapply(g, function(x) x$name) == "guide-box")]]
  lheight <- sum(legend$height)
  plt.lst <- lapply(plots, function(x) x + theme(legend.position = "none"))
  plt.lst$ncol <- ncol
  grid.arrange(
    legend,
```

```
    do.call(arrangeGrob, plt.lst),
    ncol = 1,
    heights = unit.c(lheight, unit(1, "npc") - lheight))
}

colorScorePlot <- function(Comps, Groups) {
  Comps <- unlist(Comps); Groups <- as.character(Groups)
  if (Groups == 'Gender')
    qdb <- getDB(pca.scr.df, grid.size = 25,
                 da.fit = qda.fit.gender, n.comp = Comps)
  if (Groups == 'Shoulder')
    qdb <- getDB(pca.scr.df, grid.size = 25,
                 da.fit = qda.fit.shoulder, n.comp = Comps)
  plt <- getScored(scores(pc.a), ncomp = 1:3,
                   which = Comps,
                   attr.df = attr.df,
                   col.var = Groups)
  plt <- plt + geom_contour(data = qdb,
                            aes_string(paste('PC', Comps, sep = ''),
                                       z = 'z'),
                            bins = 1)
}

plot.basis <- function(basis.obj, n = 1000){
  require(ggplot2); require(reshape2)
  get.bounds <- attr(basis.obj, 'Boundary.knots')
  x.seq <- seq(get.bounds[1], get.bounds[2], length.out = n)
  pred.bs <- predict(basis.obj, newx = x.seq)
  dt <- data.table(x = x.seq, bs = pred.bs)
  dt.stk <- melt(dt, 1, value.name = 'y', variable.name = 'BasisFun')
  plt <- ggplot(dt.stk, aes(x, y, group = BasisFun, color = BasisFun)) +
    geom_line() + theme_bw() +
    scale_color_discrete(guide = FALSE) +
    theme(axis.title = element_blank())
  return(plt)
}

plot.basisfd <- function(fd.basis.obj, n = 1000){
  require(ggplot2); require(reshape2)
  get.bounds <- fd.basis.obj$rangeval
  x.seq <- seq(get.bounds[1], get.bounds[2], length.out = n)
  # browser()
  pred.bs <- predict(fd.basis.obj, newdata = x.seq)
  dt <- data.table(x = x.seq, bs = pred.bs)
  dt.stk <- melt(dt, 1, value.name = 'y', variable.name = 'BasisFun')
  plt <- ggplot(dt.stk, aes(x, y, group = BasisFun, color = BasisFun)) +
    geom_line() + theme_bw() +
    scale_color_discrete(guide = FALSE) +
    theme(axis.title = element_blank())
  return(plt)
}
```