Assignment #4

Saied Hosseinipoor

10/31/2016

Problem 1

Problem 1 - part a

Principla Component Analysis (PCA) in R studio is performed by prcomp function. Eigen vectors are part of the results as $rotation variable. Part of the vectors are as follows:

```
##                  PC1           PC2          PC3           PC4
## pixel0  2.219274e-20 -5.732181e-19 6.287447e-20 -1.759315e-19
## pixel1  2.081668e-17  1.110223e-16 2.081668e-17  8.326673e-17
## pixel2 -1.942890e-16  0.000000e+00 4.857226e-17 -4.163336e-17
## pixel3 -1.387779e-16  1.110223e-16 4.336809e-17 -1.110223e-16
```

Problem 1 - part b

The variable $center is mean value for the data set. After Reshaphing the mean data into a 28x28 matrix results the Figure 1. It is saved as file named meanDigits.jpeg.

Problem 1 - part c

A loop has been constructed to reconstruct images #15 and #100 with lower dimensions and saves them in the separate files. The following R code was used for this part.

```r
X.mean = t(digits.mean)
for (img in c(15, 100)){                                       # Outer loop for images
  for (k in c(5, 20, 100)){                                    # Inner loop for dimensions
    X = digits.data[img,]                                      # Pick the image data
    E = digits.Eigen.Vector[,1:k]                              # Pick the eigen vectors
    weight = digits.PCA$x[img, 1:k]                            # Calulate the weight values
    new.image = X.mean + weight %*% t(E)                       # Reconstruct the image
    new.image = matrix(new.image,
                       nrow = 28, ncol=28, byrow = T)          # Converts data into a matrix to show
    image.name =
      do.call("paste0", list("image", img, "-", k, ".jpg"))    # Make proper file name
    jpeg(image.name, width = 2800, height = 2800, res = 600)  # Open a jpeg output
    image(new.image, col = grey(seq(0, 1, length = 256)))      # Write the image into a jpeg output
    dev.off()                                                  # Save the jpeg file
  }
}
```

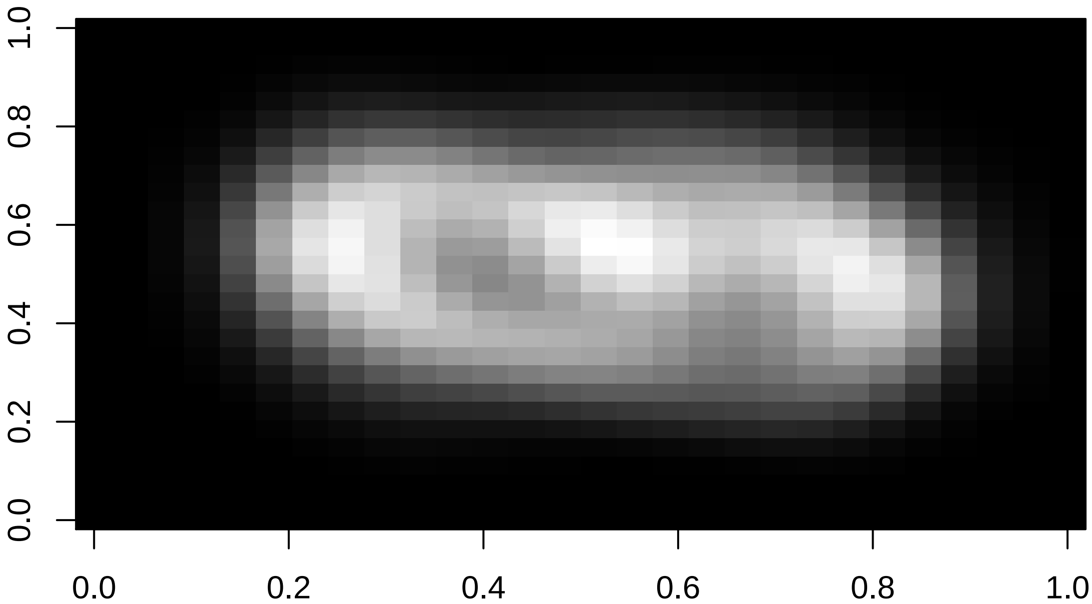

Figure 1: Mean Digit Illustration

Problem 1 - part d

There are several ways to choose pricipal components which are the dimension of the target space inder to solve the problem in less complex space. One of them is to plot the PCA and find the elbow. Other way is to consider a cut off number like 90% which is a criterion to keep a certain amount of the variances. Here the graphical way illustrated in Figure 2, but the numerical method was considered to choose proper k. To maintain 90% of the variance, 87 components have been selected.

```
s = summary(digits.PCA)$importance[3,]
names(s[s>.9][1])
```

```
## [1] "PC87"
```

To calculate the Mahalanobis distance for each point in test data set, first, the differennce between the point and mean digit space were calculated. Then the points have been projected into the new space. All the digit of digit space also projected into the new space. The Mahalanobis distances were calculated and the mean value of the distances for each point of the test data set reported.

```
k = 87                                                                   # Choose a k from above (I chose fro
X = as.matrix(class7Test[,3:786])                                        # Rest Data set
E = digits.Eigen.Vector[,1:k]                                            # Eigen vectors
X.mean = matrix(rep(digits.mean, 7),
                nrow = 7, ncol = 784, byrow = T)                         # Mean digit from digit space
X.diff = X - X.mean                                                      # Deviation form mean digit
test.projection = X.diff %*% E                                           # Map test data into new space
training.projection = digits.PCA$x[ ,1:k]                                # Map training data into the new sap
projection.corr = cov(training.projection)                               # Calculates the covariance

mahala.mean = rep(0,7)                                                   # Empty matrix
for(i in 1:7){
  mahala.mean[i] = mean(mahalanobis(                                     # Calculates the mean values for dis
```

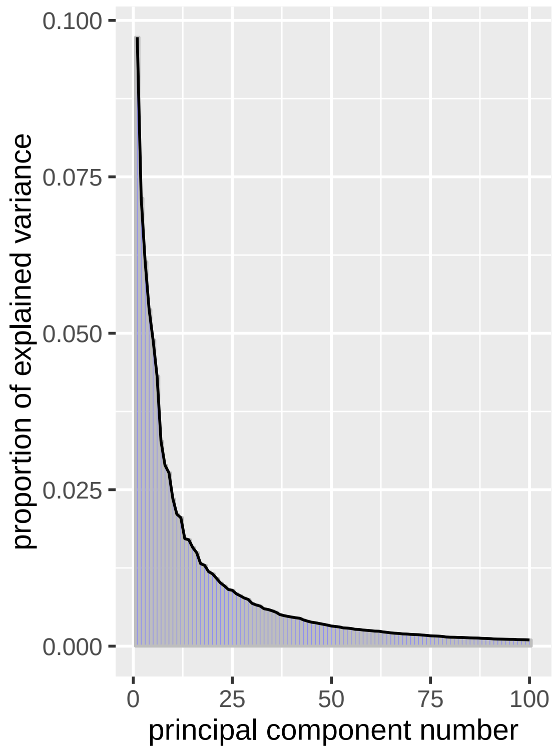

Figure 2: Mean Digit Illustration

```
    training.projection, test.projection[i,], projection.corr))     # Calculates the mah. distances
}
mahala.mean
```

```
## [1] 145.7205 162.6667 162.6691 136.8206 189.2969 213.8493 509.3984
```

Problem 1 - part d

In this section a two loops are applied to determine the minumum dimension required to recognize the digit in the test data set. The outer loop counts the image number. It starts from image number 4 to image number 6. The inner loop count the target dimension. It start with k = 1, then finds the Mahalanobis distance of the image to the all images in the digit space. The lable of the closet point is examined if it is same as our test lable, we found the correct answer, but if they are not same the loop runs for next value of the k until find the right answer.

```
for (img in 4:6){
  k = 1
  repeat {
    X = as.matrix(class7Test[,3:786])
    E = digits.Eigen.Vector[,1:k]
    X.mean = matrix(rep(t(as.matrix(digits.mean)), 7),
                    nrow = 7, ncol = 784, byrow = T)
    X.diff = X - X.mean
    test.projection = X.diff %*% E
    training.projection = as.matrix(digits.PCA$x[ ,1:k] )
```

```
    projection.corr = cov(training.projection)

    a = which.min(mahalanobis(
      training.projection, test.projection[img,], projection.corr))
    predict.lable = classDigits[a, 1]
    test.lable = class7Test[img, 2]

    if (predict.lable == test.lable || k > 784) break
    k = k + 1
  }
  print (k)
}
```

```
## [1] 4
## [1] 11
## [1] 2
```

Problem 2

Problem 2 - part a

First, the missing data and their structure was investigated, then the variables with more than 20% missing values including ***LotFrontage***, ***Alley***, ***FireplaceQu***, ***PoolQC***, ***Fence***, and ***MiscFeature*** were deleted. LotPrice variable also converted into the **log price**.

```
##  [1]   0   0   0   0 207   0 938   0   0   0   0   0   0   0   0   0   0
## [18]   0   0   0   0   0   4   4   0   0   0  31  31  32  31   0  32   0
## [35]   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
## [52]   0   0   0 466  53  53  53   0   0  53  53   0   0   0   0   0 998
## [69] 805 966   0   0   0   0   0
```

```
##  [1]  5  7 23 24 28 29 30 31 33 40 55 56 57 58 61 62 68 69 70
```

```
## [1]  5  7 55 68 69 70
```

```
## [1] "LotFrontage" "Alley"       "FireplaceQu" "PoolQC"      "Fence"
## [6] "MiscFeature"
```

In order to explore the useful variables, all the variables plotted against to the target variable and they were selected visuallu at the first step.

Some of the variables have been deleted from the data set:

To visulize the data, numeric and categorial variables have been separated to check.

```
attach(housingData.model)
numerics = sapply(housingData.model, is.numeric)
housingData.model.numerics = housingData.model[,numerics]
```

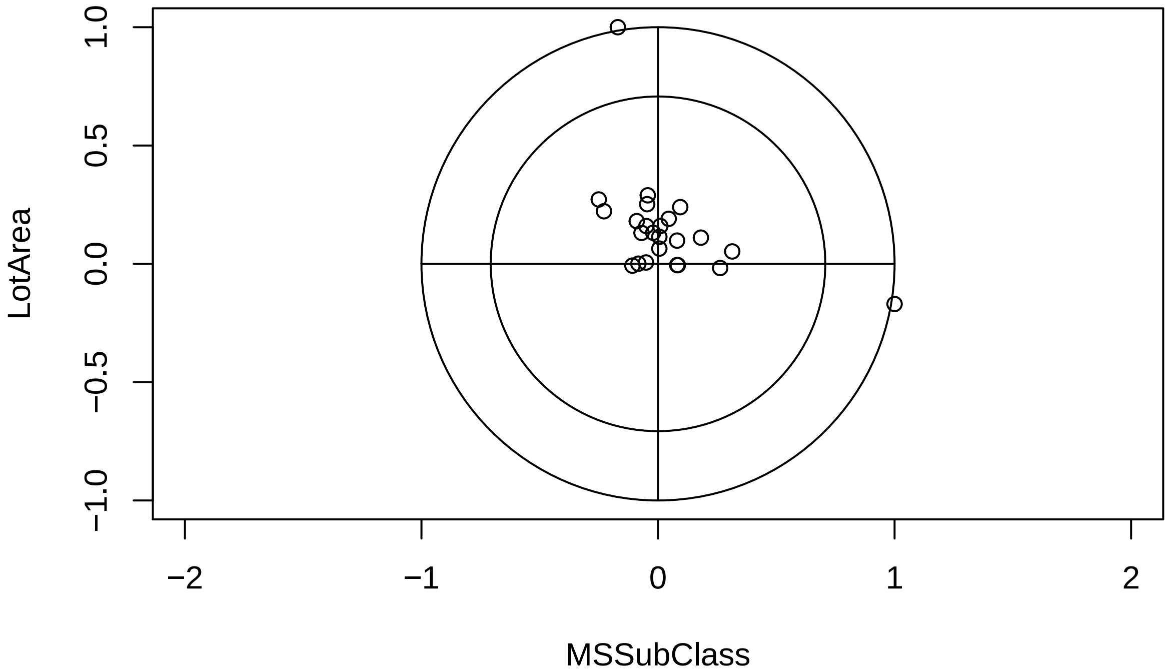

Figure 3: Predictor Correlations

```r
factors = sapply(housingData.model, is.factor)
housingData.model.factors = housingData.model[,factors]
housingData.model.factors = cbind(housingData.model.factors, housingData.model$LogSalePrice)

numericsCor = cor(housingData.model.numerics)
corrplot(numericsCor, method = "circle")
```

By checking the different paramters and drawings, some more variables were deleted. One of the important paramters was the signeifance level listed in summary and also the vif value for each variable. adj-R^2 and standard error also were considered.

```r
fitHousing = lm(LogSalePrice ~ . , data = housingData.model)
summary(fitHousing)
```

```
##
## Call:
## lm(formula = LogSalePrice ~ ., data = housingData.model)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.65849 -0.05249  0.00264  0.05818  0.26695
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          4.666e+00  6.165e-01   7.569 8.92e-14 ***
## MSSubClass          -4.709e-04  1.023e-04  -4.603 4.72e-06 ***
## LotArea              3.028e-06  4.383e-07   6.907 9.07e-12 ***
## LandSlopeMod         4.171e-03  1.571e-02   0.265 0.790708
```

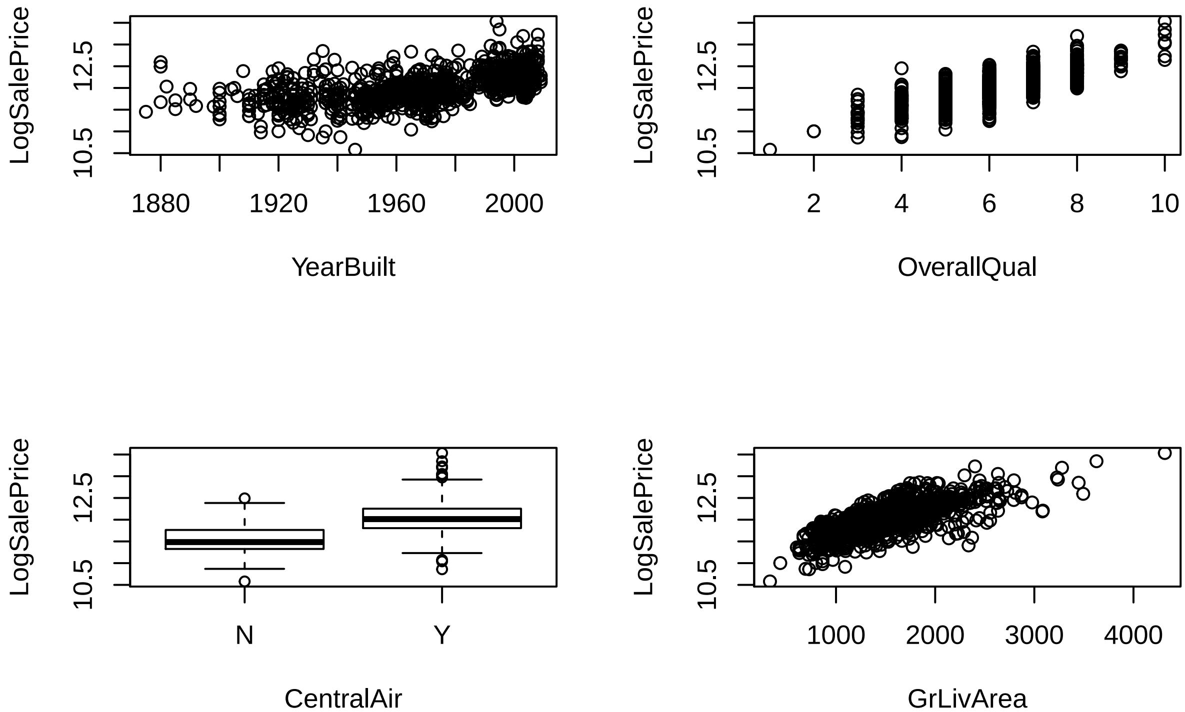

Figure 4: Predictor Correlations

```
## LandSlopeSev        -9.395e-02  5.399e-02  -1.740 0.082159 .
## NeighborhoodClearCr  4.514e-02  2.963e-02   1.523 0.128037
## NeighborhoodCollgCr -2.194e-04  2.283e-02  -0.010 0.992335
## NeighborhoodCrawfor  1.131e-01  2.491e-02   4.539 6.38e-06 ***
## NeighborhoodEdwards -4.700e-02  2.045e-02  -2.298 0.021794 *
## NeighborhoodGilbert  8.732e-03  2.550e-02   0.342 0.732074
## NeighborhoodIDOTRR  -3.788e-02  2.702e-02  -1.402 0.161207
## NeighborhoodMitchel -4.291e-02  2.485e-02  -1.727 0.084579 .
## NeighborhoodNAmes   -2.160e-02  1.913e-02  -1.129 0.259032
## NeighborhoodNoRidge -4.857e-03  2.891e-02  -0.168 0.866627
## NeighborhoodNridgHt  5.386e-02  2.829e-02   1.904 0.057178 .
## NeighborhoodNWAmes  -3.812e-02  2.347e-02  -1.624 0.104704
## NeighborhoodOldTown -7.487e-02  2.061e-02  -3.633 0.000295 ***
## Neighborhoodother   -2.889e-02  2.191e-02  -1.319 0.187523
## NeighborhoodSawyer  -4.261e-02  2.210e-02  -1.928 0.054173 .
## NeighborhoodSawyerW -3.858e-02  2.461e-02  -1.568 0.117319
## NeighborhoodSomerst  5.494e-02  2.681e-02   2.049 0.040728 *
## NeighborhoodTimber  -5.777e-03  3.150e-02  -0.183 0.854508
## Condition1Feedr      2.582e-02  2.353e-02   1.097 0.272792
## Condition1Norm       6.611e-02  1.905e-02   3.470 0.000545 ***
## Condition1PosA      -2.721e-03  4.316e-02  -0.063 0.949736
## Condition1PosN       2.428e-02  3.319e-02   0.732 0.464574
## Condition1RR         2.863e-02  2.791e-02   1.026 0.305287
## OverallQual          6.166e-02  4.320e-03  14.273  < 2e-16 ***
## OverallCond          5.086e-02  3.832e-03  13.273  < 2e-16 ***
## YearBuilt            2.725e-03  2.635e-04  10.343  < 2e-16 ***
## YearRemodAdd         2.918e-04  2.450e-04   1.191 0.233949
## BsmtFinSF1           1.717e-04  1.631e-05  10.529  < 2e-16 ***
```

```
## BsmtFinSF2          1.269e-04  2.489e-05   5.099 4.13e-07 ***
## BsmtUnfSF           9.080e-05  1.440e-05   6.305 4.41e-10 ***
## CentralAirY         3.276e-02  1.678e-02   1.952 0.051263 .
## ElectricalFuseF    -6.534e-02  2.896e-02  -2.256 0.024272 *
## ElectricalFuseP    -5.903e-02  7.291e-02  -0.810 0.418312
## ElectricalSBrkr    -1.871e-02  1.337e-02  -1.399 0.162041
## X1stFlrSF           1.516e-04  7.515e-05   2.017 0.043959 *
## X2ndFlrSF           1.449e-04  7.314e-05   1.981 0.047905 *
## GrLivArea           1.259e-04  7.211e-05   1.746 0.081116 .
## BsmtFullBath        2.927e-02  8.700e-03   3.364 0.000799 ***
## FullBath            3.247e-03  9.810e-03   0.331 0.740739
## KitchenAbvGr       -2.605e-02  1.976e-02  -1.318 0.187705
## KitchenQualAvg     -2.803e-02  9.473e-03  -2.959 0.003164 **
## KitchenQualBelowAvg -3.767e-02  2.350e-02  -1.603 0.109315
## Fireplaces          3.299e-02  6.238e-03   5.289 1.52e-07 ***
## GarageCars          4.124e-02  1.018e-02   4.050 5.55e-05 ***
## GarageArea          8.665e-05  3.554e-05   2.438 0.014949 *
## WoodDeckSF          7.618e-05  2.784e-05   2.736 0.006335 **
## OpenPorchSF         1.557e-04  5.511e-05   2.826 0.004819 **
## EncPorchSF          2.362e-04  4.212e-05   5.608 2.69e-08 ***
## SaleTypeWD          4.523e-03  1.890e-02   0.239 0.810942
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09786 on 946 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.9312, Adjusted R-squared:  0.9274
## F-statistic: 246.3 on 52 and 946 DF,  p-value: < 2.2e-16
```

```
AIC(fitHousing); BIC(fitHousing)
```

```
## [1] -1755.11
```

```
## [1] -1490.145
```

```
mean(vif(fitHousing))
```

```
## [1] 6.26058
```

```
ncvTest(fitHousing)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 21.15908    Df = 1     p = 4.226946e-06
```

```
defaultSummary(data.frame(obs=housingData.clean$LogSalePrice,pred=predict(fitHousing, housingData.clean
```

```
##       RMSE   Rsquared
## 0.09523327 0.93122954
```

At the end, a stepwise modeling using ***stepAIC*** function was examined.

```r
fitHousing.step = stepAIC(fitHousing, direction = "both", trace = FALSE)
AIC(fitHousing.step); BIC(fitHousing.step)
```

```
## [1] -1759.701
```

```
## [1] -1519.27
```

```r
mean(vif(fitHousing.step))
```

```
## [1] 6.820374
```

```r
ncvTest(fitHousing.step)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 20.35155    Df = 1     p = 6.444093e-06
```

This is the final result for stepAIC:

```r
fitHousing.step$call
```

```
## lm(formula = LogSalePrice ~ MSSubClass + LotArea + Neighborhood +
##     Condition1 + OverallQual + OverallCond + YearBuilt + BsmtFinSF1 +
##     BsmtFinSF2 + BsmtUnfSF + CentralAir + Electrical + X1stFlrSF +
##     X2ndFlrSF + GrLivArea + BsmtFullBath + KitchenAbvGr + KitchenQual +
##     Fireplaces + GarageCars + GarageArea + WoodDeckSF + OpenPorchSF +
##     EncPorchSF, data = housingData.model)
```

Problem 2 - part b

100 observations were kept for vlidation purpose. The rest of data were used to establish the model. The predictor variables were chosen based on the previous section analysis.

```r
housingData.model.b = housingData.model[101:1000,]
fitHousing.b = lm(LogSalePrice ~ ., data = housingData.model.b)
AIC(fitHousing.b); BIC(fitHousing.b)
```

```
## [1] -1590.613
```

```
## [1] -1331.344
```

```r
mean(vif(fitHousing.b))
```

```
## [1] 6.743922
```

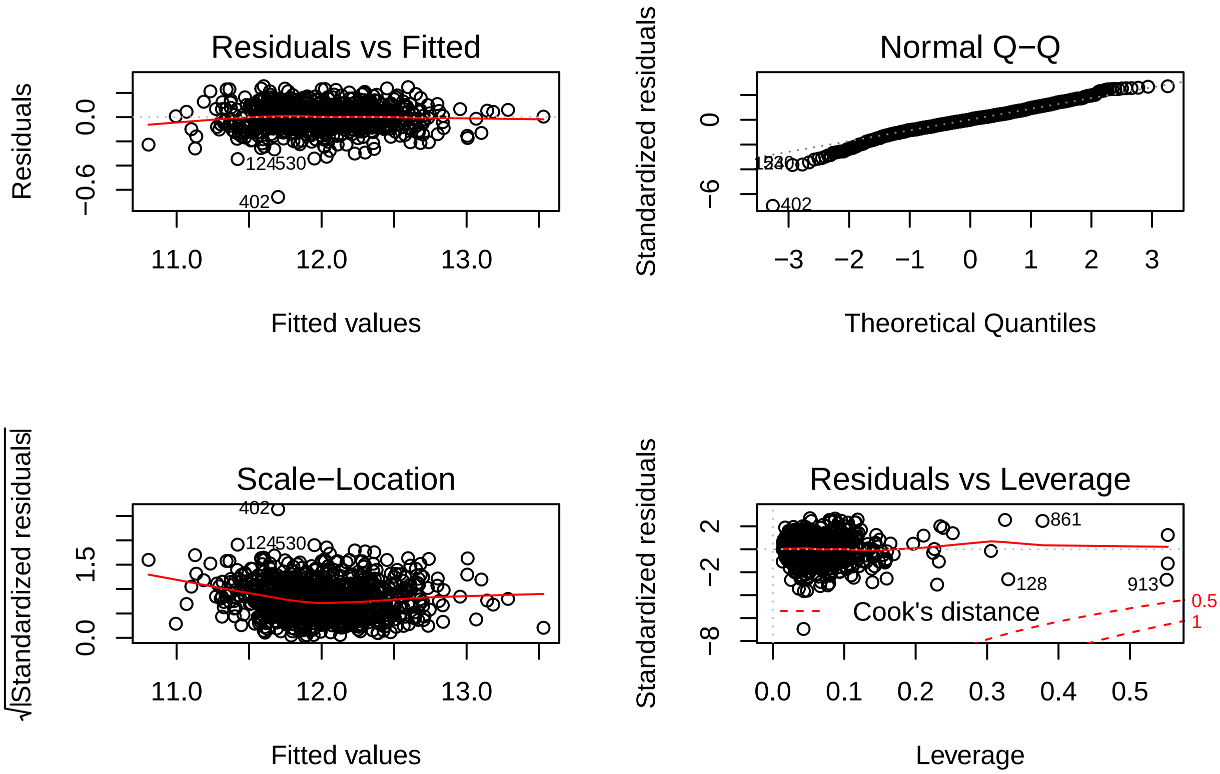

Figure 5: Residual Plots

```r
temp = anova(fitHousing.b)
par(mfrow=c(2,2)); plot(fitHousing.b); par(mfrow=c(1,1))
```

```r
ncvTest(fitHousing.b)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 10.68444    Df = 1     p = 0.001080407
```

```r
fitHousing.b$xlevels [["OverallCond"]] = union(levels(housingData.model$OverallCond),
                                               fitHousing.b$xlevels [["OverallCond"]])
predict.b = predict(fitHousing.b, housingData.model[1:100,])
defaultSummary(data.frame(obs=housingData.clean$LogSalePrice[1:100], pred = predict.b))
```

```
##      RMSE  Rsquared
## 0.1100384 0.9133391
```

Problem 2 - part c

All the data were used to build the PLS model. The following code shows the procedure and results.

```r
housingData.model.c = housingData.clean
pls.fit <- plsr(LogSalePrice~., data=housingData.model.c, validation="CV")

pls.CVRMSE <- RMSEP(pls.fit, validation = "CV")
str(pls.CVRMSE)
```

```
## List of 5
##  $ val       : num [1:2, 1, 1:153] 0.34 0.34 0.333 0.332 0.189 ...
##   ..- attr(*, "dimnames")=List of 3
##   .. ..$ estimate: chr [1:2] "CV" "adjCV"
##   .. ..$ response: chr "LogSalePrice"
##   .. ..$ model   : chr [1:153] "(Intercept)" "1 comps" "2 comps" "3 comps" ...
##  $ type      : chr "RMSEP"
##  $ comps     : num [1:153] 0 1 2 3 4 5 6 7 8 9 ...
##  $ cumulative: logi TRUE
##  $ call      : language RMSEP(object = pls.fit, validation = "CV")
##  - attr(*, "class")= chr "mvrVal"
```

```r
plot(pls.CVRMSE)
(min<-which.min(pls.CVRMSE$val[1,1,]))
```

```
## 27 comps
##       28
```

```r
points(min-1,pls.CVRMSE$val[1,1,min],col="red",cex=1.5, lwd=2)
```

```r
pls.pred <- predict(pls.fit, housingData.model.c,ncomp=1:24)
residual.c = pls.pred - housingData.model.c$LogSalePrice
```

Problem 2 - part d

A model based on LASSO was built. The foloowing code shows the procedure and results.

```r
housingData.model.d = housingData.clean
lasso.ctrl = trainControl(method="cv", number=5)
lasso.model = train(LogSalePrice ~.,
                    data = housingData.model.d, na.action = na.exclude,
                    trControl=lasso.ctrl, method="glmnet")
plot(lasso.model)
```

```r
lasso.model$bestTune
```

```
##   alpha      lambda
## 5  0.55 0.005369318
```

```r
lasso.model$resample$RMSE
```

```
## [1] 0.09086656 0.10040792 0.10481123 0.09123868 0.09466039
```

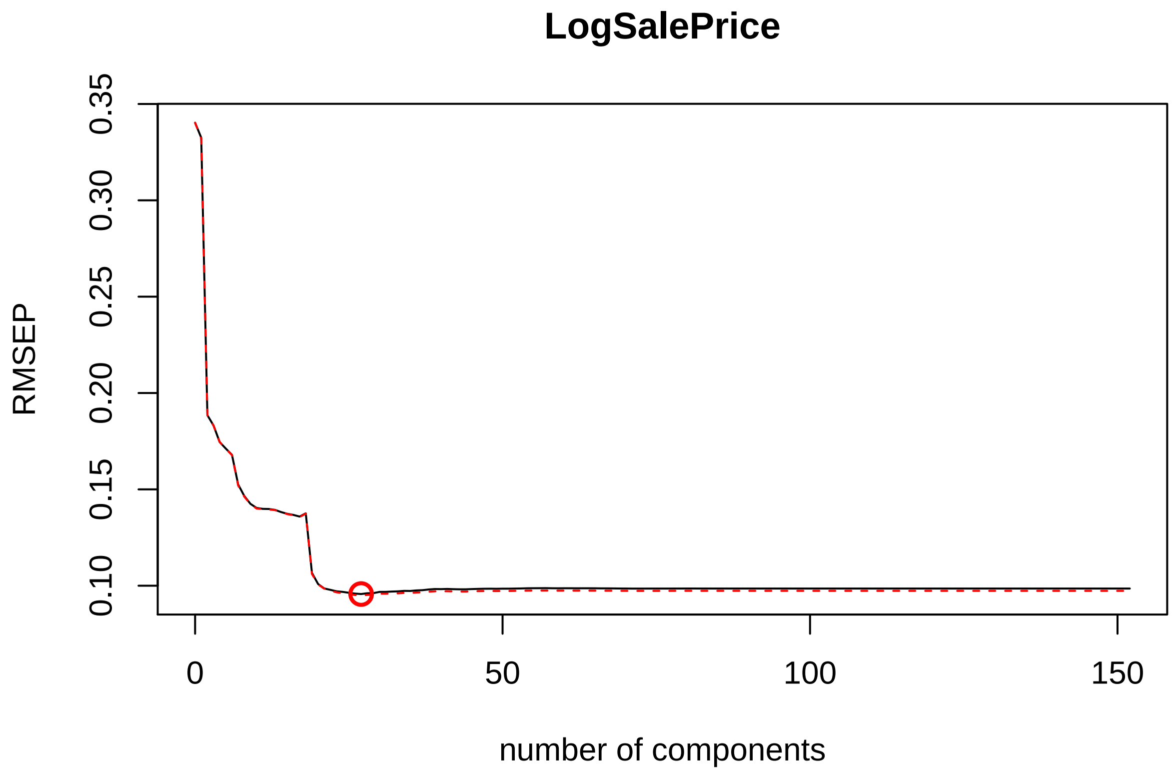

Figure 6: Component Selection in PLS Model

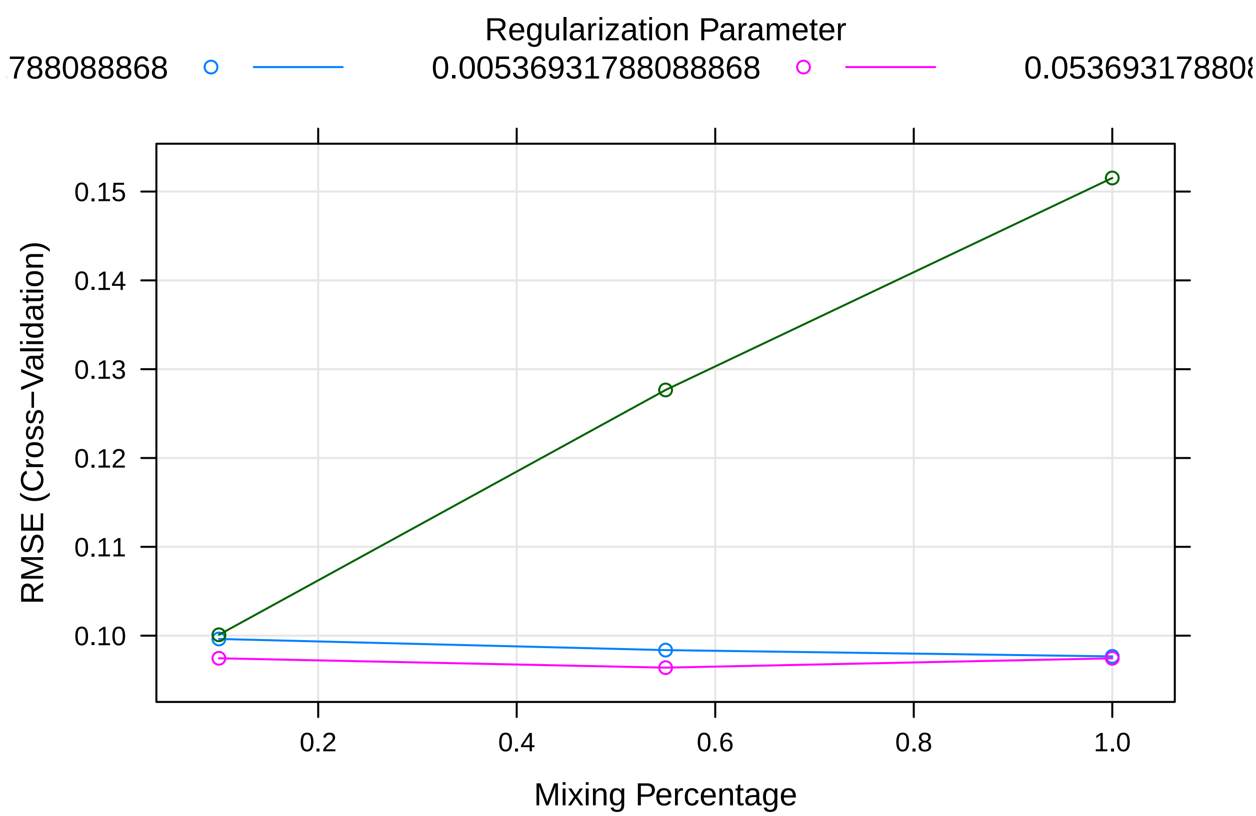

Figure 7: LASSO Model

```
lasso.model$resample$Rsquared
```

```
## [1] 0.9268780 0.9208731 0.9013699 0.9199083 0.9305610
```

```
lasso.model$results
```

```
##   alpha       lambda       RMSE  Rsquared      RMSESD RsquaredSD
## 1  0.10 0.0005369318 0.09962742 0.9159035 0.007016486 0.01003431
## 2  0.10 0.0053693179 0.09745243 0.9186845 0.006378660 0.01084675
## 3  0.10 0.0536931788 0.10009834 0.9161248 0.006392750 0.01157489
## 4  0.55 0.0005369318 0.09837264 0.9176678 0.006585969 0.01030041
## 5  0.55 0.0053693179 0.09639695 0.9199181 0.006064172 0.01125453
## 6  0.55 0.0536931788 0.12766053 0.8805623 0.007979219 0.01430991
## 7  1.00 0.0005369318 0.09766396 0.9185737 0.006406111 0.01064146
## 8  1.00 0.0053693179 0.09744820 0.9184662 0.006304339 0.01214196
## 9  1.00 0.0536931788 0.15152808 0.8461452 0.010064434 0.01766424
```

Problem 2 - part e

The linear model built on the previous sections was used to predict the given data set.

```
housingData.test = read.csv("housingTest.csv", header = TRUE)
housingData.test = housingData.test[,c(-1,-2)]

pred.e = predict(fitHousing, housingData.test)
SalesPrice = exp(pred.e)
```