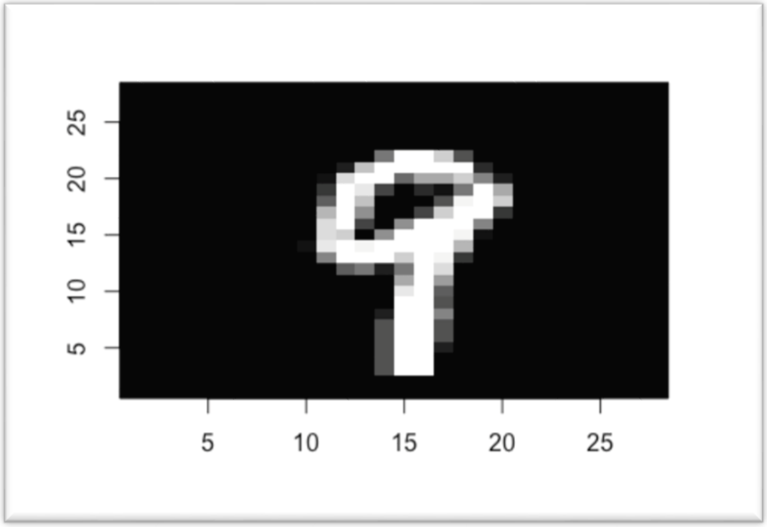
MNIST

Nikhita Kannam

The MNIST dataset is a popular dataset that contains handwritten digits as images as well as labels containing the number for those digits. The images 28x28 and the digits are 0 to 9. The dataset is often used for machine learning, classification and deep learning. The MNIST dataset has 60,000 images stored in train_digits and 10,000 labels stores in train_Labels. For this project, linear regression and logistic regression will be used on the dataset to get the Confusion Matrices, Accuracy and Error Rates. My goal is to see how often a certain digit is mistaken for another and what pairs of digits are mistaken for each other.

```
load_image_file = function(filename) {
   ret = list()
   f = file(filename, 'rb')
   readBin(f, 'integer', n = 1, size = 4, endian = 'big')
   n = readBin(f, 'integer', n = 1, size = 4, endian = 'big')
   nrow = readBin(f, 'integer', n = 1, size = 4, endian = 'big')
   ncol = readBin(f, 'integer', n = 1, size = 4, endian = 'big')
   x = readBin(f, 'integer', n = n*nrow*ncol, size = 1, signed = FALSE)
   close(f)
   data.frame(matrix(x, ncol = nrow*ncol, byrow = TRUE))
 }

load_label_file = function(filename) {
   f = file(filename, 'rb')
   readBin(f, 'integer', n = 1, size = 4, endian = 'big')
   n = readBin(f, 'integer', n = 1, size = 4, endian = 'big')
   L = readBin(f, 'integer', n = n, size = 1, signed = FALSE)
   close(f)
   L
 }

# set working directory
 #setwd("~/CSc/DSE_Applied_Stat/Project 2 Regressions")

# load images and corresponding labels
train_digits = load_image_file('train-images.idx3-ubyte')
train_Labels = load_label_file('train-labels.idx1-ubyte')

# Select digit k and re-label the dataset wrt selected digit
 k = 9
 is_k = which((train_Labels == k) %in% TRUE)
 not_k = which((train_Labels == k) %in% FALSE)

# Display i'th instance of selected digit in the training set:
 i=100
 image(1:28, 1:28, matrix(as.matrix(train_digits[is_k[i],]), nrow=28)[ , 28:1],
      col = gray(seq(0, 1, 0.05)), xlab = "", ylab="")
```



Our digit k will be 9.

We are provided the train_digits dataframe which has 60000 rows and 784 columns. Each row represents an image. The train_labels has the labels for each image. For this assignment, train_labels will be turned into a list of 1s and -1. If the label is k (in our case 9), the label will be replaced with a 1, else it will be replaced with a -1. Then train_labels will be combined with the train_digits dataframe. The result will be one dataframe that has 785 columns, the last column will be the label for the image in that row.

```
not_zero_index = which((colSums(train_digits) > 0) %in% TRUE)
```

```
#Replace train_labels with 1 if it's k and -1 if it's not k.

df_label1 <- replace(train_Labels, which((train_Labels == k) %in% TRUE), 1)
df_label1 <- replace(df_label1, which((train_Labels == k) %in% FALSE), -1)
```

Train digits has been change to remove any column which has all 0s. If the sum of all numbers in the column is less than 1, the column is removed.

```
train_digits <- train_digits[,not_zero_index]
df_y <- data.frame(df_label1)
 df <- cbind(train_digits, df_y)
```

Linear Regression

Once we have the dataframe we will perform the train and test split. For linear regression, lm() is used to fit the model.

In order to perform linear regression, the data should be split into a train and test data set. This is done with the code below. Sample splits the data in half.

fit_train will be the fit model. summary() gives the Coefficients, Residual standard error, Multiple R-squared, Adjusted R-squared, F-statistic, and p-value.

```
set.seed(1)

sample <- sample(c(TRUE, FALSE), nrow(df), replace=TRUE, prob=c(0.5,0.5))
train  <- df[sample, ]
test   <- df[!sample, ]

fit_train <- lm(train$df_label1 ~ ., data = train)
summary(fit_train)
```

```
## Call:
## lm(formula = train$df_label1 ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4137 -0.2302 -0.0494  0.1318  2.3183
##
## Coefficients: (16 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.972e-01  1.133e-02 -79.194  < 2e-16 ***
## X13         -2.173e-03  7.019e-03  -0.310 0.756917
## X14          9.814e-05  2.452e-03   0.040 0.968075
## X15                NA         NA      NA       NA
## X16                NA         NA      NA       NA
## X33                NA         NA      NA       NA
## X34         -3.046e-02  1.021e-01  -0.298 0.765325
## X35          2.242e-03  7.600e-03   0.295 0.768021
## X36         -1.344e-03  3.265e-03  -0.412 0.680602
## X37         -8.753e-05  1.324e-03  -0.066 0.947273
## X38          1.560e-04  1.387e-03   0.112 0.910475
## X39         -5.305e-04  1.148e-03  -0.462 0.643976
## X40         -1.671e-05  9.846e-04  -0.017 0.986460
## X41          1.420e-04  8.536e-04   0.166 0.867917
## X42         -3.396e-04  8.684e-04  -0.391 0.695702
## X43          2.463e-04  9.105e-04   0.271 0.786776
## X44         -2.266e-04  7.783e-04  -0.291 0.770947
## X45         -6.267e-05  7.183e-04  -0.087 0.930478
## X46         -8.277e-05  8.911e-04  -0.093 0.925992
## X47         -7.565e-04  1.033e-03  -0.732 0.463963
## X48          5.462e-04  1.935e-03   0.282 0.777712
## X49         -7.287e-04  1.760e-03  -0.414 0.678812
## X50         -2.536e-04  2.391e-03  -0.106 0.915509
## X51         -2.939e-03  6.868e-03  -0.428 0.668682
## X52          2.976e-03  5.516e-03   0.540 0.589511
## X59         -2.370e-02  5.918e-02  -0.400 0.688859
## X60          1.110e-02  5.946e-02   0.187 0.851919
## X61          8.324e-04  2.609e-03   0.319 0.749695
## X62         -1.459e-03  8.565e-03  -0.170 0.864707
## X63          1.165e-03  1.652e-03   0.705 0.480649
## X64         -4.258e-04  1.207e-03  -0.353 0.724214
## X65          2.320e-04  8.232e-04   0.282 0.778060
## X66          3.476e-04  6.136e-04   0.566 0.571077
```

```
## X67         -9.315e-05  4.667e-04  -0.200 0.841794
## X68          9.857e-06  3.873e-04   0.025 0.979696
## X69         -1.021e-04  3.280e-04  -0.311 0.755696
## X70         -5.345e-05  2.894e-04  -0.185 0.853466
## X71         -1.491e-04  2.586e-04  -0.577 0.564090
## X72          1.659e-05  2.441e-04   0.068 0.945795
## X73         -8.834e-05  2.352e-04  -0.376 0.707251
## X74         -8.800e-05  2.455e-04  -0.359 0.719945
## X75          4.421e-05  2.638e-04   0.168 0.866903
## X76         -1.213e-04  2.995e-04  -0.405 0.685453
## X77          2.958e-05  3.968e-04   0.075 0.940571
## X78          2.250e-04  5.317e-04   0.423 0.672219
## X79          2.168e-04  7.190e-04   0.302 0.763019
## X80         -1.074e-03  1.420e-03  -0.756 0.449617
## X81          8.953e-04  2.802e-03   0.320 0.749299
## X82          8.376e-04  5.262e-03   0.159 0.873540
## X87          8.587e-03  2.669e-02   0.322 0.747660
## X88         -1.796e-03  6.406e-03  -0.280 0.779221
## X89          1.799e-03  5.195e-03   0.346 0.729106
## X90          8.467e-04  1.290e-03   0.657 0.511474
## X91         -2.203e-04  8.925e-04  -0.247 0.805027
## X92          3.589e-05  6.496e-04   0.055 0.955940
## X93          1.774e-05  4.895e-04   0.036 0.971091
## X94          4.802e-05  3.586e-04   0.134 0.893474
## X95          1.449e-04  3.005e-04   0.482 0.629749
## X96          3.399e-05  2.589e-04   0.131 0.895568
## X97         -3.916e-05  2.260e-04  -0.173 0.862445
## X98         -2.985e-05  1.986e-04  -0.150 0.880517
## X99          4.478e-05  1.794e-04   0.250 0.802898
## X100        -1.562e-04  1.671e-04  -0.935 0.349981
## X101        -1.034e-04  1.656e-04  -0.624 0.532520
## X102        -3.484e-05  1.679e-04  -0.208 0.835599
## X103        -1.024e-04  1.826e-04  -0.561 0.574966
## X104        -1.288e-04  2.027e-04  -0.635 0.525137
## X105        -1.767e-05  2.496e-04  -0.071 0.943553
## X106        -1.774e-04  3.161e-04  -0.561 0.574759
## X107         7.984e-05  4.342e-04   0.184 0.854088
## X108         1.501e-04  6.366e-04   0.236 0.813615
## X109         2.553e-04  1.177e-03   0.217 0.828333
## X110        -3.738e-04  3.504e-03  -0.107 0.915043
## X111         1.683e-03  4.231e-03   0.398 0.690710
## X114                NA         NA      NA       NA
## X115        -2.193e-03  7.191e-03  -0.305 0.760401
## X116        -1.430e-03  3.188e-03  -0.449 0.653694
## X117         1.054e-03  1.039e-03   1.015 0.310092
## X118        -4.912e-04  6.133e-04  -0.801 0.423197
## X119        -6.008e-05  4.659e-04  -0.129 0.897397
## X120         1.788e-04  3.358e-04   0.533 0.594367
## X121        -3.147e-04  2.687e-04  -1.171 0.241482
## X122        -2.067e-05  2.188e-04  -0.094 0.924737
## X123        -1.367e-04  1.829e-04  -0.748 0.454640
## X124        -4.659e-06  1.537e-04  -0.030 0.975814
## X125        -1.512e-04  1.315e-04  -1.150 0.250205
## X126         1.547e-06  1.154e-04   0.013 0.989299
## X127         2.860e-05  1.045e-04   0.274 0.784304
## X128         1.010e-04  9.968e-05   1.013 0.310873
## X129         9.334e-05  9.871e-05   0.946 0.344348
## X130         1.182e-04  1.027e-04   1.151 0.249865
## X131        -1.981e-06  1.095e-04  -0.018 0.985561
## X132         1.022e-04  1.210e-04   0.845 0.397976
## X133         7.969e-05  1.410e-04   0.565 0.571931
## X134        -7.428e-05  1.718e-04  -0.432 0.665573
## X135         6.783e-05  2.190e-04   0.310 0.756798
## X136        -2.514e-04  2.888e-04  -0.871 0.383940
## X137         4.047e-04  4.351e-04   0.930 0.352278
## X138        -5.319e-04  9.410e-04  -0.565 0.571915
## X139         1.514e-03  1.679e-03   0.902 0.367214
## X140        -8.083e-03  7.850e-03  -1.030 0.303176
## X143        -1.014e-04  3.579e-03  -0.028 0.977408
## X144         2.718e-04  1.381e-03   0.197 0.844017
## X145        -1.639e-04  6.792e-04  -0.241 0.809333
```

```
## X146        -2.430e-04  4.156e-04  -0.585 0.558702
## X147        -1.334e-04  2.796e-04  -0.477 0.633183
## X148         8.479e-05  2.217e-04   0.382 0.702187
## X149         5.804e-05  1.811e-04   0.321 0.748535
## X150         1.255e-04  1.487e-04   0.844 0.398482
## X151         7.404e-05  1.270e-04   0.583 0.559836
## X152         3.718e-04  1.135e-04   3.276 0.001052 **
## X153         1.245e-04  1.031e-04   1.208 0.227231
## X154         7.947e-05  9.519e-05   0.835 0.403795
## X155        -3.477e-05  8.987e-05  -0.387 0.698827
## X156        -2.123e-04  8.674e-05  -2.447 0.014410 *
## X157        -2.091e-04  8.454e-05  -2.474 0.013377 *
## X158        -2.387e-04  8.432e-05  -2.831 0.004650 **
## X159        -1.763e-04  8.673e-05  -2.033 0.042064 *
## X160        -3.866e-05  9.397e-05  -0.411 0.680773
## X161        -9.621e-05  1.067e-04  -0.902 0.367038
## X162        -8.056e-05  1.264e-04  -0.637 0.523988
## X163         1.167e-04  1.577e-04   0.740 0.459458
## X164        -2.261e-04  2.058e-04  -1.099 0.271956
## X165         2.920e-04  2.742e-04   1.065 0.286848
## X166         2.979e-04  4.368e-04   0.682 0.495293
## X167        -7.695e-04  7.890e-04  -0.975 0.329413
## X168         9.104e-03  5.087e-03   1.790 0.073518 .
## X170        -8.149e-02  8.501e-02  -0.959 0.337780
## X171        -1.138e-04  1.890e-03  -0.060 0.951975
## X172        -5.929e-04  7.287e-04  -0.814 0.415845
## X173        -4.427e-05  4.183e-04  -0.106 0.915722
## X174         5.022e-04  2.886e-04   1.741 0.081773 .
## X175        -1.382e-05  2.155e-04  -0.064 0.948884
## X176        -5.431e-05  1.701e-04  -0.319 0.749469
## X177        -7.258e-05  1.391e-04  -0.522 0.601808
## X178         1.369e-04  1.199e-04   1.142 0.253593
## X179        -1.640e-04  1.062e-04  -1.544 0.122519
## X180         1.136e-04  9.644e-05   1.177 0.239015
## X181         2.528e-06  9.039e-05   0.028 0.977691
## X182         1.469e-04  8.782e-05   1.672 0.094462 .
## X183         8.027e-05  8.553e-05   0.938 0.348029
## X184        -2.713e-06  8.300e-05  -0.033 0.973924
## X185         1.480e-04  8.086e-05   1.830 0.067195 .
## X186         2.358e-04  8.057e-05   2.927 0.003424 **
## X187         3.363e-04  8.263e-05   4.070 4.71e-05 ***
 ## X188         2.842e-05  8.716e-05   0.326 0.744370
## X189         4.133e-05  9.807e-05   0.421 0.673432
## X190        -6.053e-05  1.144e-04  -0.529 0.596726
## X191        -1.976e-04  1.393e-04  -1.419 0.156007
## X192        -2.091e-04  1.764e-04  -1.185 0.235841
## X193        -1.307e-04  2.242e-04  -0.583 0.559914
## X194        -1.817e-04  2.961e-04  -0.614 0.539500
## X195         5.275e-04  4.864e-04   1.085 0.278147
## X196        -3.007e-03  2.149e-03  -1.400 0.161615
## X197        -5.059e-02  4.147e-01  -0.122 0.902916
## X198        -3.947e-04  2.712e-03  -0.146 0.884303
## X199        -3.961e-04  1.100e-03  -0.360 0.718820
## X200         7.009e-04  5.219e-04   1.343 0.179283
## X201        -2.165e-04  3.233e-04  -0.670 0.503056
## X202        -3.968e-04  2.308e-04  -1.719 0.085567 .
## X203        -1.762e-04  1.789e-04  -0.985 0.324662
## X204        -8.676e-06  1.435e-04  -0.060 0.951802
## X205        -1.441e-04  1.198e-04  -1.203 0.228868
## X206        -1.418e-04  1.024e-04  -1.385 0.166088
## X207        -1.403e-04  9.122e-05  -1.538 0.123984
## X208        -5.308e-05  8.581e-05  -0.619 0.536161
## X209         8.150e-05  8.365e-05   0.974 0.329923
## X210         1.068e-04  8.222e-05   1.299 0.194123
## X211         4.319e-04  8.099e-05   5.333 9.73e-08 ***
 ## X212         4.615e-04  7.982e-05   5.782 7.46e-09 ***
 ## X213         3.982e-04  7.850e-05   5.072 3.96e-07 ***
 ## X214         6.779e-05  7.846e-05   0.864 0.387591
## X215        -1.815e-04  7.968e-05  -2.277 0.022777 *
## X216        -2.063e-05  8.320e-05  -0.248 0.804181
## X217        -9.358e-05  9.360e-05  -1.000 0.317408
```

```
## X218        -1.260e-04  1.079e-04  -1.168 0.242964
## X219         5.158e-05  1.302e-04   0.396 0.691905
## X220        -7.653e-05  1.619e-04  -0.473 0.636333
## X221        -1.245e-04  1.970e-04  -0.632 0.527259
## X222         1.978e-05  2.609e-04   0.076 0.939568
## X223        -5.439e-04  4.489e-04  -1.211 0.225723
## X224         9.373e-04  1.944e-03   0.482 0.629785
## X225                NA         NA      NA       NA
## X226        -2.604e-03  2.227e-03  -1.170 0.242164
## X227        -4.111e-05  6.770e-04  -0.061 0.951582
## X228        -9.352e-04  4.269e-04  -2.190 0.028496 *
## X229         2.411e-04  2.822e-04   0.854 0.392862
## X230        -8.044e-05  2.076e-04  -0.388 0.698344
## X231        -5.221e-05  1.638e-04  -0.319 0.749983
## X232        -3.789e-04  1.312e-04  -2.887 0.003888 **
## X233         1.598e-04  1.099e-04   1.453 0.146201
## X234        -1.809e-04  9.491e-05  -1.906 0.056720 .
## X235        -1.517e-04  8.783e-05  -1.727 0.084105 .
## X236        -1.044e-04  8.331e-05  -1.253 0.210226
## X237        -6.362e-05  8.290e-05  -0.767 0.442798
## X238         2.236e-04  8.275e-05   2.702 0.006900 **
## X239         5.450e-05  8.157e-05   0.668 0.504038
## X240         4.009e-05  8.011e-05   0.501 0.616725
## X241         1.309e-04  7.835e-05   1.671 0.094736 .
## X242         6.599e-06  7.832e-05   0.084 0.932849
## X243         1.350e-04  8.005e-05   1.686 0.091733 .
## X244        -1.104e-04  8.454e-05  -1.306 0.191707
## X245        -2.866e-05  9.392e-05  -0.305 0.760226
## X246         1.210e-04  1.079e-04   1.122 0.262037
## X247        -1.942e-04  1.294e-04  -1.501 0.133431
## X248        -5.018e-05  1.611e-04  -0.311 0.755476
## X249        -3.548e-05  1.982e-04  -0.179 0.857959
## X250         2.203e-04  2.578e-04   0.854 0.392880
## X251         4.237e-04  4.795e-04   0.884 0.376888
## X252        -6.343e-04  1.578e-03  -0.402 0.687614
## X253         8.485e-02  6.285e-01   0.135 0.892604
## X254         1.067e-03  1.923e-03   0.555 0.578889
## X255        -1.187e-03  7.409e-04  -1.602 0.109192
## X256         9.127e-04  4.124e-04   2.213 0.026879 *
## X257        -2.062e-04  2.787e-04  -0.740 0.459401
## X258        -2.252e-04  2.050e-04  -1.098 0.272077
## X259         1.289e-05  1.571e-04   0.082 0.934600
## X260        -9.404e-05  1.270e-04  -0.740 0.459095
## X261        -2.562e-04  1.063e-04  -2.411 0.015918 *
## X262        -1.699e-04  9.400e-05  -1.807 0.070753 .
## X263         2.571e-05  8.781e-05   0.293 0.769717
## X264        -9.458e-06  8.472e-05  -0.112 0.911109
## X265        -3.640e-05  8.474e-05  -0.429 0.667574
## X266        -1.524e-04  8.274e-05  -1.842 0.065448 .
## X267         9.063e-06  8.186e-05   0.111 0.911838
## X268        -1.936e-05  8.135e-05  -0.238 0.811891
## X269        -1.145e-04  8.029e-05  -1.426 0.154001
## X270        -1.745e-04  8.054e-05  -2.167 0.030280 *
## X271         5.014e-05  8.266e-05   0.607 0.544110
## X272        -5.777e-05  8.749e-05  -0.660 0.509025
## X273         5.447e-05  9.672e-05   0.563 0.573350
## X274        -1.132e-04  1.127e-04  -1.004 0.315396
## X275         2.775e-05  1.330e-04   0.209 0.834762
## X276        -2.256e-04  1.642e-04  -1.374 0.169444
## X277         2.427e-05  2.225e-04   0.109 0.913118
## X278        -3.382e-04  3.003e-04  -1.126 0.259994
## X279        -2.646e-04  5.683e-04  -0.466 0.641540
## X280         7.146e-04  1.697e-03   0.421 0.673776
## X281        -7.443e-02  5.404e-01  -0.138 0.890455
## X282        -2.013e-03  1.371e-03  -1.468 0.142057
## X283         6.902e-04  7.451e-04   0.926 0.354258
## X284        -4.761e-04  4.215e-04  -1.130 0.258623
## X285        -4.572e-04  2.862e-04  -1.598 0.110149
## X286         4.092e-05  2.051e-04   0.200 0.841868
## X287        -1.754e-05  1.576e-04  -0.111 0.911366
## X288        -1.506e-04  1.263e-04  -1.192 0.233090
```

```
## X289         1.673e-04  1.061e-04   1.577 0.114849
## X290         1.725e-04  9.354e-05   1.844 0.065135 .
## X291         5.900e-05  8.800e-05   0.670 0.502564
## X292         1.334e-04  8.551e-05   1.561 0.118630
## X293        -1.908e-05  8.306e-05  -0.230 0.818296
## X294        -1.054e-04  8.376e-05  -1.258 0.208312
## X295        -1.650e-04  8.499e-05  -1.941 0.052260 .
## X296        -4.143e-05  8.468e-05  -0.489 0.624658
## X297         3.787e-06  8.304e-05   0.046 0.963631
## X298         1.211e-04  8.175e-05   1.481 0.138634
## X299        -2.662e-05  8.442e-05  -0.315 0.752473
## X300         1.009e-05  8.905e-05   0.113 0.909800
## X301        -6.894e-05  9.933e-05  -0.694 0.487623
## X302        -5.638e-05  1.157e-04  -0.487 0.626060
## X303        -2.053e-05  1.401e-04  -0.147 0.883480
## X304         1.415e-04  1.784e-04   0.793 0.427686
## X305        -1.451e-04  2.503e-04  -0.579 0.562269
## X306        -2.982e-04  3.683e-04  -0.810 0.418134
## X307         1.603e-04  6.946e-04   0.231 0.817520
## X308        -3.037e-03  1.997e-03  -1.521 0.128293
## X309                NA         NA      NA       NA
## X310         1.842e-03  1.509e-03   1.221 0.222018
## X311        -1.020e-03  8.181e-04  -1.247 0.212384
## X312        -7.098e-05  4.775e-04  -0.149 0.881819
## X313         2.377e-04  3.101e-04   0.767 0.443247
## X314         5.372e-05  2.122e-04   0.253 0.800172
## X315        -1.399e-04  1.602e-04  -0.873 0.382613
## X316         6.785e-05  1.268e-04   0.535 0.592637
## X317         2.234e-04  1.062e-04   2.104 0.035359 *
## X318        -5.883e-06  9.405e-05  -0.063 0.950129
## X319         1.509e-04  8.797e-05   1.716 0.086189 .
## X320         1.800e-05  8.552e-05   0.210 0.833341
## X321         2.417e-05  8.403e-05   0.288 0.773659
## X322         7.670e-05  8.416e-05   0.911 0.362103
## X323        -1.371e-04  8.581e-05  -1.598 0.110126
## X324         1.959e-04  8.569e-05   2.286 0.022236 *
## X325        -7.425e-05  8.342e-05  -0.890 0.373417
## X326         2.716e-05  8.191e-05   0.332 0.740212
## X327         9.778e-05  8.563e-05   1.142 0.253493
## X328         1.296e-04  9.104e-05   1.423 0.154723
## X329         2.197e-04  1.011e-04   2.172 0.029830 *
## X330         1.396e-04  1.204e-04   1.159 0.246342
## X331         2.821e-05  1.509e-04   0.187 0.851732
## X332        -7.237e-05  1.957e-04  -0.370 0.711586
## X333         1.970e-04  2.826e-04   0.697 0.485724
## X334        -6.471e-04  5.174e-04  -1.251 0.211103
## X335        -4.829e-04  8.145e-04  -0.593 0.553247
## X336         6.626e-03  6.099e-03   1.087 0.277248
## X337                NA         NA      NA       NA
## X338        -2.761e-03  2.070e-03  -1.334 0.182241
## X339         9.837e-04  1.042e-03   0.944 0.344952
## X340        -6.675e-04  5.567e-04  -1.199 0.230459
## X341        -8.154e-04  3.360e-04  -2.427 0.015243 *
## X342         7.367e-05  2.144e-04   0.344 0.731180
## X343         3.912e-04  1.604e-04   2.439 0.014734 *
## X344         1.030e-04  1.267e-04   0.813 0.416378
## X345         1.367e-06  1.062e-04   0.013 0.989733
## X346         2.064e-04  9.342e-05   2.209 0.027160 *
## X347         1.035e-04  8.833e-05   1.172 0.241402
## X348        -1.691e-05  8.415e-05  -0.201 0.840743
## X349        -4.151e-05  8.361e-05  -0.497 0.619545
## X350        -4.235e-05  8.475e-05  -0.500 0.617258
## X351         1.839e-05  8.698e-05   0.211 0.832548
## X352         6.010e-05  8.481e-05   0.709 0.478559
## X353         1.495e-04  8.167e-05   1.831 0.067103 .
## X354         7.030e-05  8.039e-05   0.874 0.381902
## X355        -1.128e-05  8.373e-05  -0.135 0.892788
## X356        -1.231e-05  9.019e-05  -0.136 0.891452
## X357         7.149e-05  1.014e-04   0.705 0.480775
## X358         8.611e-05  1.226e-04   0.702 0.482387
## X359         3.251e-04  1.587e-04   2.048 0.040576 *
```

```
## X360        4.470e-04  2.104e-04   2.124 0.033642 *
## X361        3.247e-04  2.942e-04   1.104 0.269761
## X362       -2.984e-04  6.117e-04  -0.488 0.625639
## X363       -1.231e-03  9.574e-04  -1.286 0.198412
## X364       -3.465e-03  4.446e-03  -0.779 0.435810
## X365               NA         NA      NA       NA
## X366        4.234e-03  4.678e-03   0.905 0.365433
## X367       -2.774e-03  1.498e-03  -1.852 0.064056 .
## X368        4.676e-04  7.084e-04   0.660 0.509208
## X369        5.727e-04  3.411e-04   1.679 0.093169 .
## X370        4.349e-04  2.176e-04   1.998 0.045694 *
## X371       -1.159e-04  1.579e-04  -0.734 0.462928
## X372        1.834e-04  1.242e-04   1.477 0.139741
## X373        2.120e-04  1.036e-04   2.045 0.040821 *
## X374        2.747e-05  9.171e-05   0.300 0.764523
## X375       -4.195e-05  8.613e-05  -0.487 0.626167
## X376       -7.390e-05  8.409e-05  -0.879 0.379489
## X377       -4.291e-05  8.306e-05  -0.517 0.605464
## X378       -3.573e-05  8.506e-05  -0.420 0.674430
## X379        1.351e-04  8.615e-05   1.569 0.116756
## X380        9.792e-05  8.185e-05   1.196 0.231584
## X381        1.428e-04  7.819e-05   1.827 0.067754 .
## X382        1.767e-05  7.942e-05   0.222 0.823935
## X383        2.745e-04  8.208e-05   3.344 0.000827 ***
## X384        1.778e-04  8.864e-05   2.006 0.044829 *
## X385        1.678e-07  1.011e-04   0.002 0.998675
## X386        9.314e-05  1.247e-04   0.747 0.454972
## X387       -4.902e-05  1.576e-04  -0.311 0.755827
## X388       -1.319e-04  2.125e-04  -0.621 0.534874
## X389       -2.354e-04  2.841e-04  -0.828 0.407466
## X390       -4.290e-04  6.031e-04  -0.711 0.476839
## X391        4.635e-04  1.054e-03   0.440 0.660105
## X392        1.071e-03  1.847e-03   0.580 0.562204
## X393       -3.370e-03  4.003e-03  -0.842 0.399959
## X394       -3.594e-03  5.792e-03  -0.621 0.534893
## X395       -2.915e-04  2.423e-03  -0.120 0.904249
## X396        1.301e-03  7.882e-04   1.650 0.098933 .
## X397       -2.140e-04  3.373e-04  -0.635 0.525717
## X398       -6.937e-05  2.017e-04  -0.344 0.730912
## X399        3.524e-04  1.491e-04   2.364 0.018092 *
## X400       -1.752e-04  1.196e-04  -1.464 0.143157
## X401       -9.344e-05  1.007e-04  -0.928 0.353548
## X402        1.495e-04  8.991e-05   1.662 0.096471 .
## X403        1.077e-04  8.521e-05   1.264 0.206331
## X404        1.071e-04  8.363e-05   1.281 0.200148
## X405        1.700e-04  8.366e-05   2.032 0.042197 *
## X406       -6.764e-05  8.634e-05  -0.783 0.433375
## X407        2.536e-04  8.529e-05   2.973 0.002951 **
## X408       -6.677e-05  7.926e-05  -0.842 0.399522
## X409        4.915e-05  7.654e-05   0.642 0.520781
## X410        2.239e-04  7.943e-05   2.819 0.004822 **
## X411       -1.873e-04  8.099e-05  -2.313 0.020728 *
## X412        3.022e-05  8.826e-05   0.342 0.732100
## X413       -4.089e-05  1.021e-04  -0.401 0.688767
## X414       -3.005e-05  1.253e-04  -0.240 0.810462
## X415       -5.134e-05  1.578e-04  -0.325 0.744857
## X416       -6.254e-05  2.053e-04  -0.305 0.760632
## X417       -1.304e-04  2.778e-04  -0.469 0.638948
## X418        2.394e-04  5.260e-04   0.455 0.649073
## X419       -7.068e-04  1.039e-03  -0.680 0.496511
## X420       -4.561e-04  1.182e-02  -0.039 0.969228
## X421               NA         NA      NA       NA
## X422        9.200e-02  1.248e-01   0.737 0.460898
## X423       -8.626e-04  2.126e-03  -0.406 0.684899
## X424       -2.057e-03  7.503e-04  -2.742 0.006107 **
## X425        6.922e-04  3.177e-04   2.179 0.029336 *
## X426       -3.065e-04  1.869e-04  -1.640 0.101089
## X427       -4.512e-05  1.413e-04  -0.319 0.749504
## X428        1.055e-04  1.162e-04   0.908 0.364006
## X429        1.562e-04  1.000e-04   1.561 0.118443
## X430        2.944e-05  9.091e-05   0.324 0.746061
```

```
## X431        -2.316e-05  8.680e-05  -0.267 0.789623
## X432         1.122e-05  8.581e-05   0.131 0.895940
## X433         1.016e-05  8.592e-05   0.118 0.905911
## X434         3.386e-05  8.681e-05   0.390 0.696545
## X435        -9.540e-05  8.354e-05  -1.142 0.253497
## X436        -1.302e-04  7.719e-05  -1.687 0.091621 .
## X437         1.528e-04  7.746e-05   1.972 0.048583 *
## X438         1.319e-04  7.981e-05   1.652 0.098462 .
## X439         5.795e-06  8.205e-05   0.071 0.943696
## X440        -1.065e-04  9.004e-05  -1.182 0.237073
## X441        -5.583e-05  1.043e-04  -0.535 0.592406
## X442        -9.517e-06  1.263e-04  -0.075 0.939950
## X443        -1.864e-04  1.546e-04  -1.206 0.227943
## X444        -1.628e-04  1.948e-04  -0.836 0.403432
## X445         1.535e-04  2.695e-04   0.569 0.569040
## X446        -1.215e-04  4.786e-04  -0.254 0.799560
## X447         5.468e-04  9.849e-04   0.555 0.578740
## X448        -1.117e-03  2.644e-03  -0.423 0.672585
## X449               NA         NA      NA       NA
## X450        -2.411e-03  4.928e-03  -0.489 0.624665
## X451         1.016e-03  2.036e-03   0.499 0.617902
## X452         8.198e-04  6.628e-04   1.237 0.216181
## X453        -3.242e-04  2.774e-04  -1.169 0.242609
## X454         1.211e-04  1.756e-04   0.690 0.490378
## X455        -1.097e-04  1.375e-04  -0.798 0.424955
## X456        -1.676e-04  1.137e-04  -1.474 0.140612
## X457        -5.032e-05  1.008e-04  -0.499 0.617646
## X458         4.508e-05  9.261e-05   0.487 0.626398
## X459         2.064e-04  8.963e-05   2.303 0.021272 *
## X460         6.044e-05  8.882e-05   0.680 0.496211
## X461         1.580e-04  8.811e-05   1.793 0.072928 .
## X462        -1.332e-04  8.665e-05  -1.538 0.124122
## X463        -1.177e-04  8.121e-05  -1.450 0.147111
## X464         1.078e-05  7.907e-05   0.136 0.891536
## X465         1.486e-04  7.977e-05   1.863 0.062519 .
## X466        -2.949e-05  8.091e-05  -0.365 0.715462
## X467        -1.263e-04  8.375e-05  -1.508 0.131626
## X468        -8.003e-05  9.230e-05  -0.867 0.385907
## X469        -1.626e-04  1.056e-04  -1.540 0.123607
## X470        -1.971e-04  1.266e-04  -1.557 0.119472
## X471         1.962e-04  1.535e-04   1.278 0.201289
## X472        -1.835e-04  1.926e-04  -0.953 0.340579
## X473        -1.690e-04  2.481e-04  -0.681 0.495701
## X474         3.778e-04  4.146e-04   0.911 0.362279
## X475        -4.185e-04  8.744e-04  -0.479 0.632213
## X476        -1.035e-04  2.068e-03  -0.050 0.960072
## X478        -1.374e-02  1.195e-02  -1.150 0.250181
## X479        -7.062e-04  1.743e-03  -0.405 0.685342
## X480        -1.150e-04  5.425e-04  -0.212 0.832081
## X481        -2.806e-04  2.497e-04  -1.124 0.261103
## X482        -2.553e-05  1.646e-04  -0.155 0.876754
## X483         4.092e-05  1.358e-04   0.301 0.763079
## X484         1.070e-04  1.143e-04   0.937 0.348952
## X485         6.014e-05  1.023e-04   0.588 0.556643
## X486        -1.183e-04  9.393e-05  -1.260 0.207727
## X487         9.987e-06  9.226e-05   0.108 0.913795
## X488         4.027e-05  9.135e-05   0.441 0.659357
## X489         2.234e-05  8.976e-05   0.249 0.803418
## X490         5.682e-05  8.626e-05   0.659 0.510122
## X491        -2.777e-04  8.111e-05  -3.424 0.000618 ***
 ## X492         7.616e-05  7.972e-05   0.955 0.339468
## X493        -2.360e-05  8.118e-05  -0.291 0.771246
## X494        -1.259e-05  8.260e-05  -0.152 0.878860
## X495        -3.754e-05  8.727e-05  -0.430 0.667055
## X496        -1.074e-04  9.551e-05  -1.125 0.260799
## X497         8.416e-05  1.084e-04   0.776 0.437485
## X498        -5.520e-05  1.264e-04  -0.437 0.662406
## X499        -1.051e-04  1.550e-04  -0.678 0.497698
## X500         4.407e-07  1.903e-04   0.002 0.998152
## X501        -1.943e-05  2.418e-04  -0.080 0.935955
## X502        -2.138e-04  3.512e-04  -0.609 0.542605
```

```
## X503         3.589e-04  7.573e-04   0.474 0.635556
## X504         2.497e-04  3.836e-03   0.065 0.948105
## X505        -2.404e-03  7.541e-03  -0.319 0.749885
## X506         1.517e-02  9.789e-03   1.549 0.121285
## X507         1.127e-04  1.119e-03   0.101 0.919792
## X508         3.791e-04  4.604e-04   0.823 0.410253
## X509        -1.683e-05  2.335e-04  -0.072 0.942521
## X510        -1.436e-04  1.613e-04  -0.891 0.373094
## X511        -3.669e-05  1.340e-04  -0.274 0.784224
## X512        -6.284e-05  1.135e-04  -0.554 0.579747
## X513        -1.112e-04  1.018e-04  -1.093 0.274341
## X514        -8.020e-06  9.522e-05  -0.084 0.932880
## X515        -2.161e-05  9.393e-05  -0.230 0.818070
## X516         2.734e-05  9.185e-05   0.298 0.765959
## X517        -9.261e-05  8.978e-05  -1.031 0.302321
## X518        -1.197e-04  8.639e-05  -1.385 0.166021
## X519         1.194e-04  8.313e-05   1.436 0.150910
## X520         1.141e-04  8.117e-05   1.405 0.159898
## X521        -1.147e-04  8.123e-05  -1.412 0.157984
## X522         3.324e-05  8.375e-05   0.397 0.691397
## X523        -1.495e-04  8.983e-05  -1.665 0.095997 .
## X524        -6.790e-05  9.776e-05  -0.695 0.487318
## X525        -1.927e-05  1.105e-04  -0.174 0.861578
## X526        -3.616e-05  1.284e-04  -0.282 0.778152
## X527         9.836e-05  1.557e-04   0.632 0.527697
## X528        -7.834e-05  1.931e-04  -0.406 0.685031
## X529        -9.022e-05  2.476e-04  -0.364 0.715579
## X530         3.339e-04  3.610e-04   0.925 0.355039
## X531        -2.905e-04  8.873e-04  -0.327 0.743382
## X532        -1.532e-03  3.664e-03  -0.418 0.675849
## X533               NA         NA      NA       NA
## X534        -5.397e-03  3.690e-03  -1.462 0.143638
## X535        -6.384e-04  9.638e-04  -0.662 0.507732
## X536        -3.664e-04  3.969e-04  -0.923 0.355827
## X537         1.884e-04  2.218e-04   0.849 0.395669
## X538         4.985e-05  1.588e-04   0.314 0.753665
## X539        -4.170e-05  1.326e-04  -0.315 0.753113
## X540        -4.773e-06  1.124e-04  -0.042 0.966140
## X541         3.683e-05  1.006e-04   0.366 0.714163
## X542        -2.016e-04  9.407e-05  -2.143 0.032156 *
## X543        -2.027e-04  9.191e-05  -2.206 0.027404 *
## X544        -1.066e-04  9.007e-05  -1.183 0.236786
## X545        -9.005e-05  8.807e-05  -1.022 0.306592
## X546        -5.412e-05  8.509e-05  -0.636 0.524741
## X547        -1.460e-04  8.336e-05  -1.752 0.079852 .
## X548        -2.258e-05  8.200e-05  -0.275 0.783050
## X549        -5.585e-06  8.270e-05  -0.068 0.946162
## X550        -1.693e-04  8.624e-05  -1.963 0.049645 *
## X551         6.228e-05  9.126e-05   0.682 0.495003
## X552        -8.569e-07  9.938e-05  -0.009 0.993120
## X553        -9.937e-05  1.135e-04  -0.875 0.381408
## X554         1.001e-05  1.346e-04   0.074 0.940702
## X555         7.736e-05  1.616e-04   0.479 0.632174
## X556         1.970e-05  2.037e-04   0.097 0.922942
## X557        -7.297e-05  2.716e-04  -0.269 0.788223
## X558        -3.926e-04  4.206e-04  -0.933 0.350622
## X559         7.413e-05  1.049e-03   0.071 0.943684
## X560         1.858e-03  5.225e-03   0.356 0.722057
## X562         5.177e-04  5.948e-03   0.087 0.930643
## X563         4.177e-04  8.437e-04   0.495 0.620533
## X564        -1.567e-04  3.765e-04  -0.416 0.677401
## X565        -2.537e-04  2.230e-04  -1.138 0.255320
## X566        -3.208e-05  1.568e-04  -0.205 0.837938
## X567         2.091e-05  1.278e-04   0.164 0.870025
## X568        -2.138e-05  1.115e-04  -0.192 0.847941
## X569        -1.364e-04  9.919e-05  -1.375 0.169133
## X570         2.384e-05  9.144e-05   0.261 0.794286
## X571        -4.529e-05  8.709e-05  -0.520 0.603036
## X572        -5.104e-05  8.590e-05  -0.594 0.552387
## X573        -2.211e-04  8.540e-05  -2.589 0.009617 **
## X574         1.246e-04  8.388e-05   1.486 0.137364
```

```
## X575        -9.674e-05  8.246e-05  -1.173 0.240752
## X576        -4.171e-06  8.238e-05  -0.051 0.959619
## X577         2.818e-05  8.379e-05   0.336 0.736617
## X578        -2.056e-05  8.807e-05  -0.234 0.815369
## X579        -6.171e-05  9.444e-05  -0.653 0.513487
## X580         3.516e-05  1.037e-04   0.339 0.734597
## X581         7.646e-06  1.222e-04   0.063 0.950116
## X582        -4.310e-06  1.451e-04  -0.030 0.976306
## X583        -1.034e-04  1.796e-04  -0.576 0.564855
## X584         8.388e-05  2.304e-04   0.364 0.715814
## X585        -1.025e-05  2.969e-04  -0.035 0.972473
## X586         5.629e-04  4.849e-04   1.161 0.245713
## X587        -7.105e-04  1.300e-03  -0.546 0.584810
## X588        -6.547e-04  6.393e-03  -0.102 0.918440
## X589                NA         NA      NA       NA
## X590        -2.536e-04  8.648e-03  -0.029 0.976606
## X591        -2.241e-04  8.692e-04  -0.258 0.796507
## X592        -5.989e-05  3.955e-04  -0.151 0.879637
## X593         2.482e-04  2.334e-04   1.063 0.287568
## X594        -8.677e-05  1.608e-04  -0.540 0.589513
## X595        -4.062e-06  1.300e-04  -0.031 0.975073
## X596        -5.640e-05  1.108e-04  -0.509 0.610874
## X597         3.951e-05  9.999e-05   0.395 0.692749
## X598        -6.072e-05  9.170e-05  -0.662 0.507919
## X599        -4.975e-05  8.711e-05  -0.571 0.567967
## X600        -1.066e-04  8.393e-05  -1.271 0.203905
## X601         1.044e-05  8.267e-05   0.126 0.899459
## X602        -2.088e-04  8.352e-05  -2.500 0.012417 *
## X603         2.357e-05  8.313e-05   0.284 0.776771
## X604        -1.615e-04  8.409e-05  -1.921 0.054747 .
## X605        -4.083e-05  8.658e-05  -0.472 0.637199
## X606        -8.529e-05  9.176e-05  -0.929 0.352648
## X607        -6.757e-06  1.011e-04  -0.067 0.946704
## X608        -3.970e-05  1.140e-04  -0.348 0.727741
## X609         5.043e-05  1.342e-04   0.376 0.707169
## X610         3.583e-05  1.643e-04   0.218 0.827379
## X611        -8.660e-05  2.101e-04  -0.412 0.680253
## X612         1.869e-04  2.769e-04   0.675 0.499581
## X613        -1.755e-04  3.560e-04  -0.493 0.622011
## X614        -3.517e-04  5.853e-04  -0.601 0.547957
## X615        -1.900e-03  1.478e-03  -1.285 0.198655
## X616        -1.417e-03  1.162e-02  -0.122 0.902994
## X617                NA         NA      NA       NA
## X618         2.275e-02  2.049e-02   1.110 0.266957
## X619        -1.702e-04  1.153e-03  -0.148 0.882649
## X620        -2.832e-04  4.749e-04  -0.596 0.550966
## X621         7.458e-05  2.719e-04   0.274 0.783912
## X622         6.837e-06  1.835e-04   0.037 0.970277
## X623        -9.418e-06  1.398e-04  -0.067 0.946292
## X624         7.565e-05  1.162e-04   0.651 0.514959
## X625        -1.198e-04  1.010e-04  -1.186 0.235631
## X626         9.154e-06  9.306e-05   0.098 0.921647
## X627        -5.340e-05  8.882e-05  -0.601 0.547709
## X628         1.507e-05  8.670e-05   0.174 0.861999
## X629        -4.981e-05  8.544e-05  -0.583 0.559920
## X630        -9.063e-05  8.496e-05  -1.067 0.286112
## X631        -1.037e-04  8.561e-05  -1.212 0.225644
## X632        -1.087e-04  8.866e-05  -1.226 0.220087
## X633         4.775e-05  9.240e-05   0.517 0.605334
## X634         5.284e-05  1.005e-04   0.526 0.599102
## X635        -2.117e-05  1.124e-04  -0.188 0.850663
## X636         1.009e-04  1.321e-04   0.764 0.444977
## X637        -2.284e-04  1.621e-04  -1.409 0.158837
## X638         3.499e-04  2.026e-04   1.727 0.084235 .
## X639        -3.343e-04  2.607e-04  -1.283 0.199649
## X640        -4.999e-05  3.466e-04  -0.144 0.885306
## X641         8.768e-04  4.849e-04   1.808 0.070587 .
## X642        -1.686e-03  9.593e-04  -1.757 0.078896 .
## X643         8.217e-03  7.331e-03   1.121 0.262339
## X644                NA         NA      NA       NA
## X647         7.528e-05  1.777e-03   0.042 0.966204
```

```
## X648         4.368e-04  6.263e-04   0.697 0.485503
## X649        -3.454e-04  3.588e-04  -0.963 0.335724
## X650         6.814e-05  2.313e-04   0.295 0.768257
## X651         2.289e-05  1.671e-04   0.137 0.891012
## X652        -3.789e-04  1.316e-04  -2.879 0.003997 **
## X653         1.897e-04  1.104e-04   1.718 0.085827 .
## X654        -1.787e-04  9.927e-05  -1.800 0.071804 .
## X655        -1.387e-04  9.158e-05  -1.514 0.129974
## X656        -1.173e-04  8.761e-05  -1.339 0.180617
## X657        -1.514e-04  8.667e-05  -1.746 0.080738 .
## X658        -5.210e-05  8.604e-05  -0.606 0.544830
## X659        -1.054e-04  8.699e-05  -1.211 0.225779
## X660        -2.471e-04  9.146e-05  -2.702 0.006895 **
## X661        -1.176e-04  1.007e-04  -1.167 0.243192
## X662        -2.124e-04  1.138e-04  -1.866 0.061993 .
## X663        -2.128e-04  1.349e-04  -1.577 0.114805
## X664         6.679e-06  1.625e-04   0.041 0.967206
## X665        -1.991e-04  2.058e-04  -0.967 0.333400
## X666        -2.538e-04  2.682e-04  -0.946 0.344066
## X667         6.227e-04  3.511e-04   1.773 0.076182 .
## X668        -2.214e-04  4.787e-04  -0.462 0.643785
## X669        -1.472e-04  7.012e-04  -0.210 0.833689
## X670        -7.443e-04  1.512e-03  -0.492 0.622633
## X671        -2.634e-03  2.533e-03  -1.040 0.298309
## X675         1.200e-03  3.968e-03   0.303 0.762240
## X676        -1.620e-04  1.047e-03  -0.155 0.877083
## X677        -4.222e-04  5.765e-04  -0.732 0.463992
## X678         1.309e-04  3.761e-04   0.348 0.727842
## X679        -3.417e-04  2.521e-04  -1.356 0.175259
## X680         4.679e-05  1.846e-04   0.254 0.799872
## X681        -8.114e-05  1.503e-04  -0.540 0.589346
## X682         7.880e-06  1.270e-04   0.062 0.950518
## X683        -2.007e-04  1.110e-04  -1.808 0.070623 .
## X684        -1.788e-05  1.001e-04  -0.179 0.858234
## X685        -1.016e-04  9.803e-05  -1.036 0.300201
## X686        -7.489e-05  9.726e-05  -0.770 0.441305
## X687        -9.536e-05  9.951e-05  -0.958 0.337954
## X688        -3.552e-05  1.059e-04  -0.335 0.737302
## X689        -1.541e-04  1.194e-04  -1.290 0.197104
## X690         5.301e-05  1.388e-04   0.382 0.702511
## X691        -4.897e-05  1.670e-04  -0.293 0.769331
## X692        -7.040e-05  2.077e-04  -0.339 0.734713
## X693         3.703e-04  2.684e-04   1.379 0.167757
## X694        -2.319e-04  3.497e-04  -0.663 0.507158
## X695         2.813e-04  5.270e-04   0.534 0.593523
## X696         1.382e-03  7.605e-04   1.818 0.069107 .
## X697         1.307e-03  1.266e-03   1.032 0.302141
## X698        -1.485e-03  2.532e-03  -0.587 0.557471
## X699         2.290e-02  2.921e-02   0.784 0.433051
## X703        -3.874e-03  2.365e-02  -0.164 0.869909
## X704         4.916e-04  2.224e-03   0.221 0.825062
## X705        -2.326e-03  1.024e-03  -2.271 0.023147 *
## X706         1.402e-03  6.240e-04   2.247 0.024640 *
## X707         1.913e-04  4.099e-04   0.467 0.640760
## X708         1.559e-04  2.877e-04   0.542 0.587880
## X709         2.144e-04  2.142e-04   1.001 0.316943
## X710        -8.269e-05  1.749e-04  -0.473 0.636420
## X711         9.921e-04  1.517e-04   6.540 6.24e-11 ***
## X712        -1.791e-05  1.384e-04  -0.129 0.897024
## X713         8.080e-04  1.293e-04   6.249 4.18e-10 ***
## X714         5.695e-04  1.298e-04   4.387 1.16e-05 ***
## X715         1.869e-04  1.340e-04   1.395 0.163099
## X716         8.540e-04  1.446e-04   5.905 3.57e-09 ***
## X717         7.857e-04  1.620e-04   4.849 1.24e-06 ***
## X718         1.218e-03  1.862e-04   6.545 6.05e-11 ***
## X719         1.252e-03  2.196e-04   5.700 1.21e-08 ***
## X720         1.618e-03  2.881e-04   5.616 1.97e-08 ***
## X721         1.518e-03  3.926e-04   3.867 0.000110 ***
## X722         2.869e-03  5.090e-04   5.637 1.75e-08 ***
## X723        -8.468e-04  7.508e-04  -1.128 0.259378
## X724         4.094e-03  1.243e-03   3.293 0.000994 ***
```

```
## X725        -6.163e-03  2.106e-03  -2.926 0.003437 **
## X726         1.775e-02  5.289e-03   3.356 0.000792 ***
## X727        -4.405e-02  2.749e-02  -1.602 0.109090
## X732         1.633e-02  1.560e-02   1.047 0.295073
## X733         4.802e-03  1.843e-03   2.606 0.009168 **
## X734        -2.013e-03  1.061e-03  -1.898 0.057718 .
## X735         4.809e-04  5.750e-04   0.836 0.402949
## X736        -1.921e-04  4.184e-04  -0.459 0.646164
## X737         1.553e-04  3.065e-04   0.507 0.612458
## X738        -7.392e-05  2.498e-04  -0.296 0.767327
## X739        -1.782e-04  2.117e-04  -0.841 0.400117
## X740        -3.202e-04  1.906e-04  -1.680 0.093003 .
## X741         1.581e-04  1.803e-04   0.877 0.380413
## X742        -2.186e-04  1.819e-04  -1.202 0.229293
## X743         6.001e-04  1.885e-04   3.183 0.001459 **
## X744        -2.867e-05  2.078e-04  -0.138 0.890271
## X745        -4.544e-04  2.331e-04  -1.949 0.051275 .
## X746         8.186e-04  2.752e-04   2.975 0.002935 **
## X747        -6.339e-04  3.420e-04  -1.854 0.063814 .
## X748        -6.608e-05  4.625e-04  -0.143 0.886404
## X749         5.570e-04  6.214e-04   0.896 0.370137
## X750        -1.211e-03  8.717e-04  -1.389 0.164728
## X751        -2.213e-03  1.699e-03  -1.302 0.192892
## X752         6.345e-03  3.309e-03   1.918 0.055156 .
## X753        -2.894e-01  1.878e-01  -1.541 0.123225
## X754        -2.752e-04  7.777e-03  -0.035 0.971773
## X761                NA         NA      NA       NA
## X762         9.074e-03  9.345e-03   0.971 0.331551
## X763        -3.298e-03  2.913e-03  -1.132 0.257576
## X764         1.775e-03  1.654e-03   1.073 0.283275
## X765        -1.282e-03  1.265e-03  -1.013 0.310953
## X766         6.474e-04  7.845e-04   0.825 0.409271
## X767        -3.818e-04  6.615e-04  -0.577 0.563778
## X768         1.229e-04  6.019e-04   0.204 0.838215
## X769        -5.530e-04  4.798e-04  -1.153 0.249124
## X770        -9.167e-04  4.925e-04  -1.861 0.062708 .
## X771        -1.357e-04  4.823e-04  -0.281 0.778442
## X772        -1.047e-03  5.581e-04  -1.876 0.060711 .
## X773         4.830e-05  6.231e-04   0.078 0.938220
## X774        -1.848e-03  6.817e-04  -2.711 0.006715 **
## X775        -9.158e-04  8.808e-04  -1.040 0.298456
## X776         9.524e-04  1.619e-03   0.588 0.556446
## X777        -5.261e-03  2.572e-03  -2.045 0.040815 *
## X778         1.342e-02  8.474e-03   1.584 0.113214
## X779        -2.602e-02  1.357e-02  -1.918 0.055145 .
## X780                NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4506 on 29347 degrees of freedom
## Multiple R-squared:  0.4462, Adjusted R-squared:  0.433
## F-statistic: 33.74 on 701 and 29347 DF,  p-value: < 2.2e-16
```

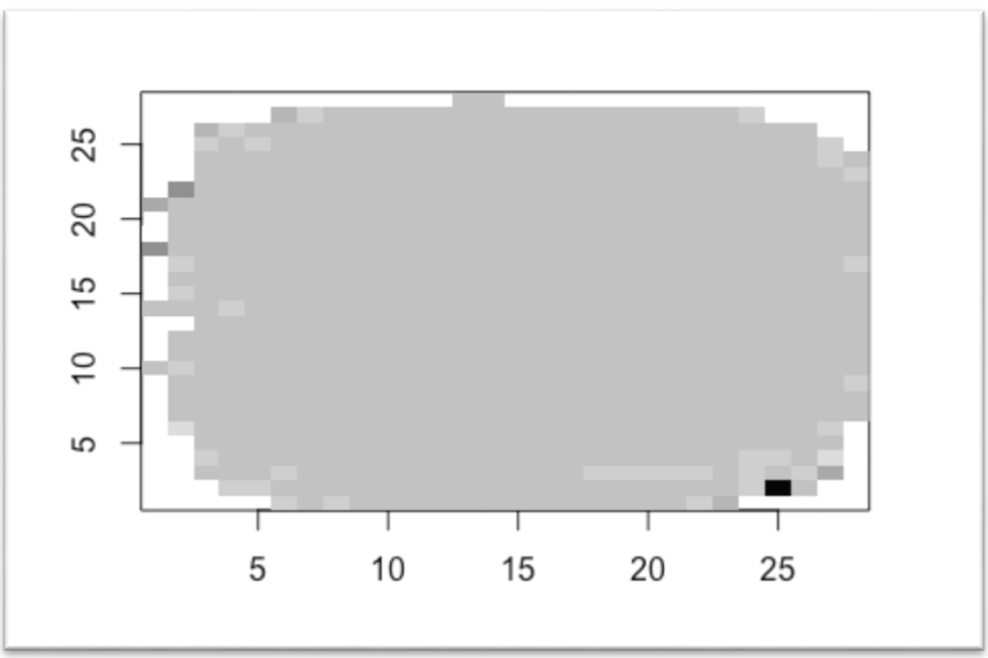
Beta Image

The columns with all 0s have been removed. There are no longer 784 columns in our train_digits dataset. This means our image is smaller than a 28x28 matrix. However, we need a 28x28 matrix for the beta image. Below is the code to pass in a 28x28 matrix into image().

First an empty list is created to hold the coefficients at the right index. The coefficients are placed in a one column dataframe called beta_image. The for loop will go through not_zero_index, which is a list of indices where the sum of a column is **not** equal to 0. The coefficient will be placed at the correct index. The list is then converted into a 28x28 matrix which is passed into image() to output the beta image.

```
set.seed(1)
 empty_list_beta <- vector("list", 784)
 beta_image <- data.frame(fit_train$coefficients)
 i <- 2
 for (num in not_zero_index){

  empty_list_beta[num] <- beta_image[i,]
   i <- i+1
 }
```

```
empty_matrix_beta <- matrix(empty_list_beta, nrow = 28, ncol = 28)
empty_matrix_beta <- replace(empty_matrix_beta, empty_matrix_beta== 'NULL', NA)


image(1:28, 1:28, matrix(as.matrix(as.numeric(empty_matrix_beta)), nrow=28)[ ,
28:1],
      col = gray(seq(0, 1, 0.05)), xlab = "", ylab="")
```



Confusion matrix for test and train sets

```
predicted <- predict(fit_train, test)
p_class <- ifelse(predicted >.5,"1","-1")
confusion_mat_test <- table(p_class, test[['df_label1']])
confusion_mat_test
```

```
  p_class     -1      1
       -1 26942   2771
       1      47   191
```

```
predicted <- predict(fit_train, train, type="response")
p_class <- ifelse(predicted >.5,"1","-1")
confusion_mat_train <- table(p_class, train[['df_label1']])
confusion_mat_train
```

```
p_class     -1      1
       -1 27038   2790
        1     24   197
```

Accuracy and Classification Error rate

Formula for Accuracy: (TP + TN) / (TP + FP + TN + FN)

Formula for Classification Error rate: 1 - Accuracy

```
#For the testing set
TP <- confusion_mat_test[1,1]
TN <- confusion_mat_test[2,2]
FP <- confusion_mat_test[1,2]
FN <- confusion_mat_test[2,1]
Accuracy_test <- (TP + TN) / (TP + FP + TN + FN)
classification_error_test <- 1-Accuracy_test
paste0('The Accuracy for the testing set is: ',Accuracy_test)
paste0('The classification error rate for the testing set is: 
',classification_error_test)
```

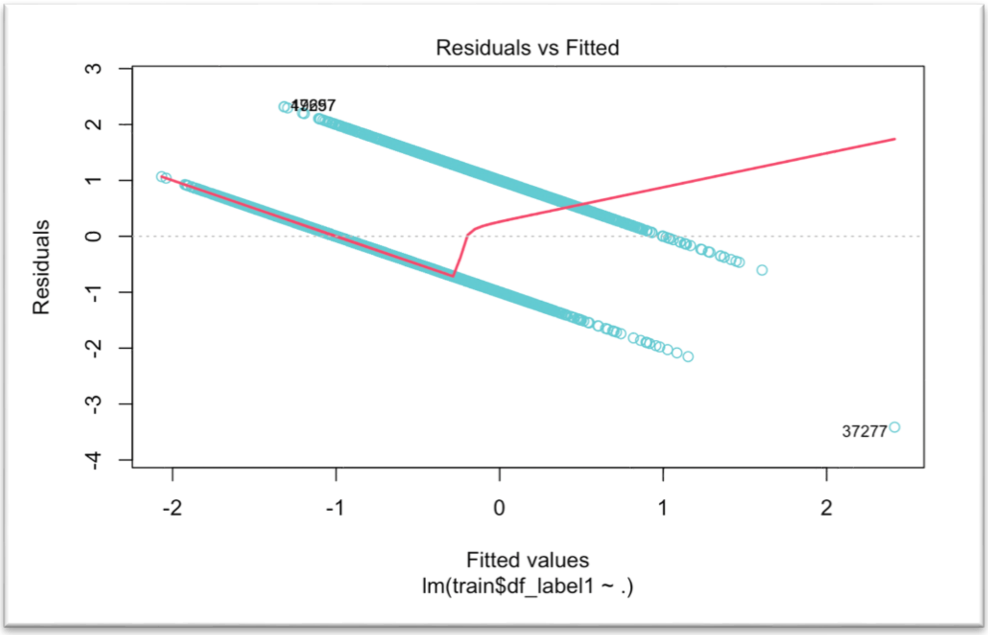
```
"The Accuracy for the testing set is: 0.905912991218991"

"The classification error rate for the testing set is: 
0.094087008781009"
```

```
TP <- confusion_mat_train[1,1]
TN <- confusion_mat_train[2,2]
FP <- confusion_mat_train[1,2]
FN <- confusion_mat_train[2,1]
Accuracy <- (TP + TN) / (TP + FP + TN + FN)
classification_error_train <- 1-Accuracy
paste0('The Accuracy for the training set is: ',Accuracy_test)
paste0('The classification error rate for the training set is: 
',classification_error_train)
```

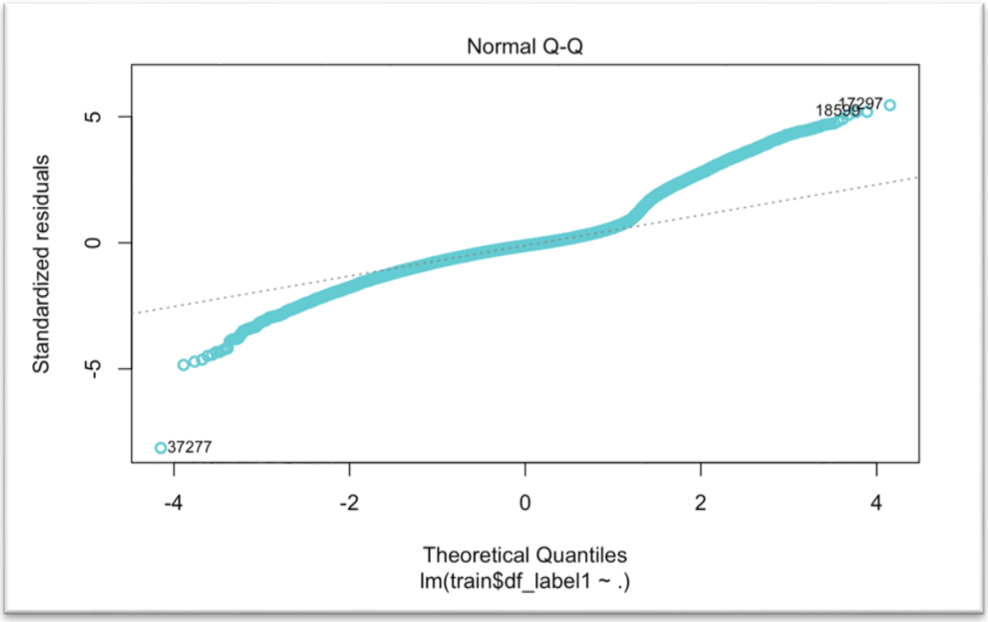
```
"The Accuracy for the training set is: 0.905912991218991"

"The classification error rate for the training set is: 
0.093647043162834"
```

The accuracy and error rate for both training and testing are very similar. For the testing set, the error rate is 0.094087 and for the training set the error rate is 0.0936. The error rate is slightly higher for the testing set which can be expected because we fit the training set.

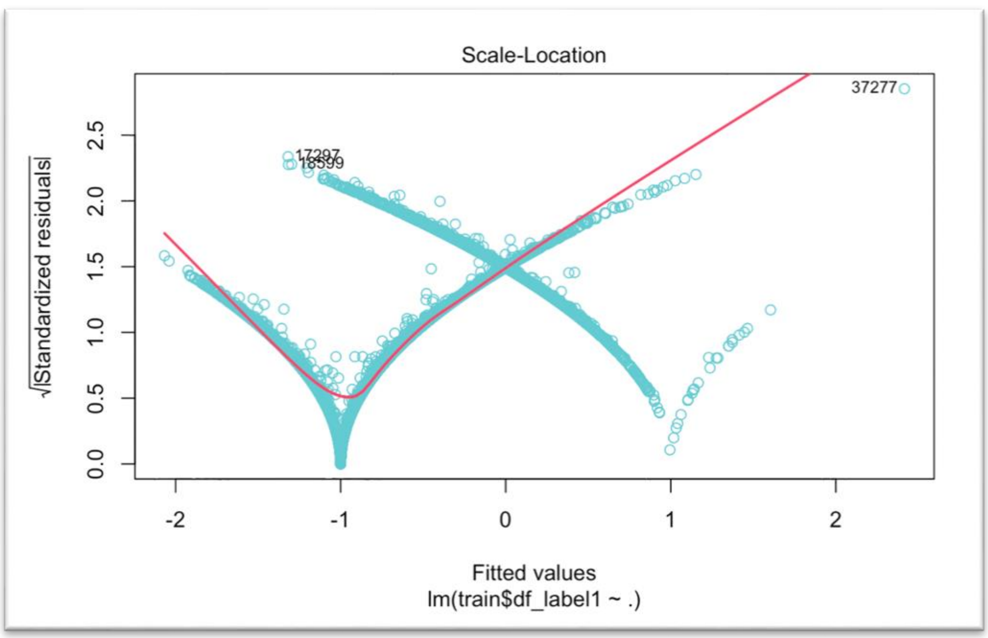
```
plot(fit_train, lwd = '2', col = 'cadetblue3')
```



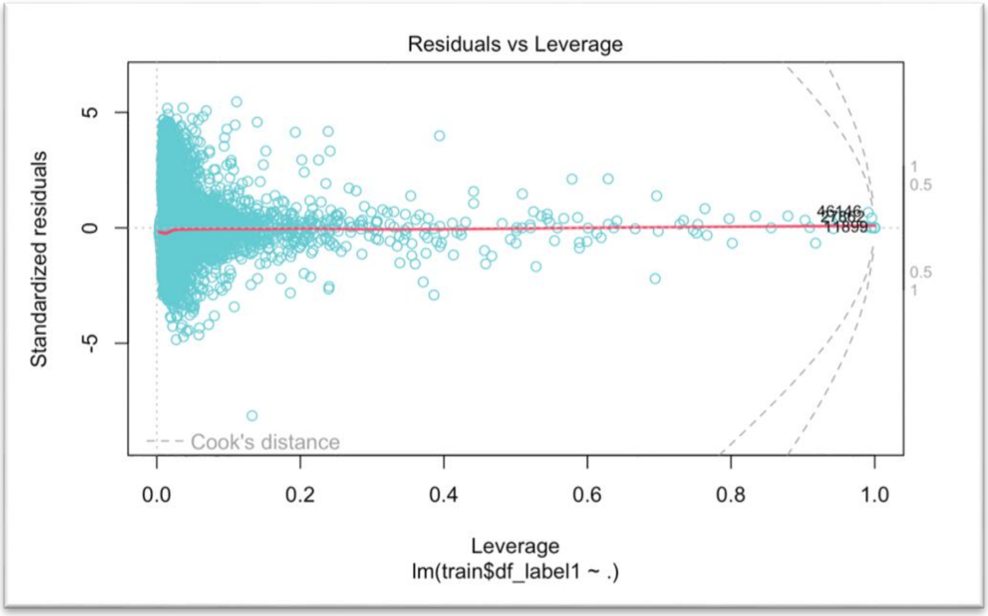
The residual and fitted plot have scatterplots that seem to be in 2 straight lines. The number of points above residual 0 and the number below seem to be around the same. There are a couple points that stand out, meaning points stray from the rest of the points. This means that there are a few outliers.



QQ plots can help us understand the distribution of the data. Here the points are mostly in a straight line in the middle but curve at both ends with a big jump at Quantile 2. The data exhibits that it might have more extreme values than what would be expected in a Normal Distribution.



Due to the red line not going horizontally with, it can be assumed that homoscedasticity is not satisfies for the regression model. Meaning the spread of residuals is not equal at all fitted values.



The leverage is the extent to which the coefficients in the modal would change if that particular data point was removed. There are a few points that are close to the cooks distance lines (the gray dashed lines) however it does not appear that any points are inside these lines. So, it can be assumed that there aren't any significantly influential points. However, some points come very close.

Part 2

Picking 2 digits

Instead of using just one digit and comparing it to the others. We will use two digits. For example, if our digits are 0 and 1. Images that are labelled 0 and 1 are kept, the label 0 will be changed to 1 and 1 will be changed to -1. Same with 02, 03, etc.

The code used in part 1 is put under 2 for loops. The first for loop will keep track of digit k and the second for loop will keep track of the second digit. In this chunk of code, list_df holds the label column (1 or –1) for each pair. List_of_df_lin holds the dataframes after the label column is combined.

```
#Use the same logic used in Part 1. Have lists to keep track of pairs, lists and
dataframes. label = list()
 list_df = list()
 list_of_pairs = list()


 #Use for loop. The first digit will be x
 for (x in 0:9){
   corr = x
```

```
  list_to_nine <- list(0,1,2,3,4,5,6,7,8,9)
  second_loop_list <- list_to_nine[-c(corr+1)]
  #Second digit will be y


  for (y in 0:9) {
    #y should greater than x because we don't need pairs like 11, 22. We also don't
need pairs twice, for example 12 and 21 are the same pair
    if (y > x) {
      incorr = y
      pair = list()
      pair <- append(pair,corr)
      pair <- append(pair,incorr)
      #store the pair in a list
      list_of_pairs[[length(list_of_pairs) + 1]] <- pair
      #corr is one digit and incorr is the second
      df_label2 <- replace(train_Labels, which((train_Labels == corr) %in% TRUE), 1)
      df_label2 <- replace(df_label2, which((train_Labels == incorr) %in% TRUE), -1)
      list_df[[length(list_df) + 1]] <- df_label2
    }
  }
 }


#This list will store all the new dataframes for each digit.
list_of_df_lin = list()
 for (one_pair in list_df){
  df_y2 <- data.frame(one_pair)
  df2 <- cbind(train_digits, df_y2)
  df2 <- subset(df2,(one_pair==1|one_pair==-1))
  list_of_df_lin[[length(list_of_df_lin) + 1]] <- df2

}
```

This chunk of code is calculating the confusion matrix, accuracy and error rates for each of the new dataframes.

```
set.seed(1)
 #Error Rates for Train and Error Rates for Test will be stored in these lists.
train_error <- list()
 test_error <- list()
 for (df_for_split in list_of_df_lin){

  sample <- sample(c(TRUE, FALSE), nrow(df_for_split), replace=TRUE, prob=c(0.5,0.5))
   train01  <- df_for_split[sample, ]
   test01   <- df_for_split[!sample, ]
   fit_train01 <- lm(train01$one_pair ~ ., data = train01)

  predicted01 <- predict(fit_train01, test01, type="response")
   p_class01 <- ifelse(predicted01 >.5,"1","-1")
   confusion_mat_test <- table(p_class01, test01[['one_pair']])


  predicted <- predict(fit_train01, train01, type="response")
  p_class <- ifelse(predicted >.5,"1","-1")
  confusion_mat_train <- table(p_class, train01[['one_pair']])
```

```
  TP <- confusion_mat_test[1,1]
  TN <- confusion_mat_test[2,2]
  FP <- confusion_mat_test[1,2]
  FN <- confusion_mat_test[2,1]
  Accuracy_test <- (TP + TN) / (TP + FP + TN + FN)
  classification_error_test <- 1-Accuracy_test
  test_error[[length(test_error) + 1]] <- classification_error_test



  TP <- confusion_mat_train[1,1]
  TN <- confusion_mat_train[2,2]
  FP <- confusion_mat_train[1,2]
  FN <- confusion_mat_train[2,1]
  Accuracy <- (TP + TN) / (TP + FP + TN + FN)
  classification_error_train <- 1-Accuracy
  train_error[[length(train_error) + 1]] <- classification_error_train

}
```

Matrix with error rates

Right now there are 45 error rates for the testing sets and 45 error rates for the training sets. The test error rates will be displayed on the lower half of the matrix and the train error rates will be on the upper half. A 10x10 matrix is created and lower.tri and upper.tri are used to fill in the matrix. The diagonal will be empty because those are the pairs of 11, 22, etc.

```
suppressWarnings({
 my_mat <- matrix(, ncol = 10, nrow=10)

my_mat[lower.tri(my_mat, diag = FALSE)] <- test_error[1:45]
my_mat <- matrix(my_mat, ncol = 10, nrow=10)
my_mat[upper.tri(my_mat, diag = FALSE)] <- train_error[1:45]
my_mat <- rbind(c(0:9), my_mat)
my_mat <- cbind(c(-1:9), my_mat)
my_mat
})
```

-1	0	1	2	3	4	5	6	7	8	9
0	NA	0.005890782	0.05316154	0.02604222	0.02123613	0.01953304	0.0410937	0.03674022	0.09060655	0.03840378
1	0.01315789	NA	0.04534031	0.06515455	0.07843137	0.01029963	0.0127349	0.07960576	0.04124044	0.1128205
2	0.06434431	0.02796104	NA	0.04314913	0.02353957	0.0170227	0.06250661	0.03202656	0.04351204	0.04625429
3	0.05721393	0.02531646	0.07828498	NA	0.02083991	0.01175561	0.04133218	0.03271875	0.02459802	0.01414141
4	0.03607236	0.02040155	0.05673531	0.04350622	NA	0.01487911	0.04889329	0.08298034	0.04188206	0.04819407
5	0.07358117	0.02398315	0.05685841	0.09552107	0.05146732	NA	0.04442049	0.03229614	0.05450549	0.01530558
6	0.05119966	0.02130751	0.05180349	0.04054482	0.02996378	0.0529794	NA	0.04276453	0.07055777	0.04686843
7	0.03486529	0.02395118	0.04755798	0.05598496	0.05506937	0.04656271	0.02605042	NA	0.03854699	0.07213115
8	0.0916532	0.04858811	0.09261644	0.1014986	0.07391537	0.1295887	0.05662625	0.05587669	NA	0.04836461
9	0.0341067	0.02095329	0.04252121	0.05380178	0.0830095	0.0630303	0.02331103	0.08409641	0.0629272	NA

The ***highest*** error rate is ***0.1295887*** for the pair ***5*** and ***8*** in the testing data. People might have more of a chance of confusing these 2 digits because the lower half of both digits curves the same way. 5 is very similar to 8 except it's not completely connected at the top and bottom. The second highest error rate was 3 and 8. This could be because the right side of both digits are the same. Since 5 and 8 are mixed up and 3 and 8 are mixed up often, I was interested to see the error rate for 3 and 5. 3 and 5 have the third highest error rate in the testing set.

The **lowest** error rate is **0.005890782** for the pair **0** and **1**. 0 and 1 do not look similar. This is also the case for the training set.

However, for the training set the highest error rate is .1128 for the digits 1 and 9.

Beta image for the highest error rate

```
#The pair with the highest error rate is 5 and 8
 df_label58 <- replace(train_Labels, which((train_Labels == 5) %in% TRUE), 1)
 df_label58 <- replace(df_label58, which((train_Labels == 8) %in% TRUE), -1)

df_y58 <- data.frame(df_label58)
df58 <- cbind(train_digits, df_y58)
df58 <- subset(df58,(df_label58==1|df_label58==-1))

set.seed(1)

sample <- sample(c(TRUE, FALSE), nrow(df58), replace=TRUE, prob=c(0.5,0.5))
train58  <- df58[sample, ]
test58  <- df58[!sample, ]
fit_train58 <- lm(train58$df_label58 ~ ., data = train58)

set.seed(1)
empty_list_beta58 <- vector("list", 784)
beta_image58 <- data.frame(fit_train58$coefficients)
i <- 2
for (num58 in not_zero_index){

  empty_list_beta58[num58] <- beta_image58[i,]
  i <- i+1

}

empty_matrix_beta58 <- matrix(empty_list_beta58, nrow = 28, ncol = 28)
empty_matrix_beta58 <- replace(empty_matrix_beta58, empty_matrix_beta58== 'NULL', NA)
image(1:28, 1:28, matrix(as.matrix(as.numeric(empty_matrix_beta58)), nrow=28)[ , 28:1],
      col = gray(seq(0, 1, 0.05)), xlab = "", ylab="")
```

Beta Image for the lowest error rate

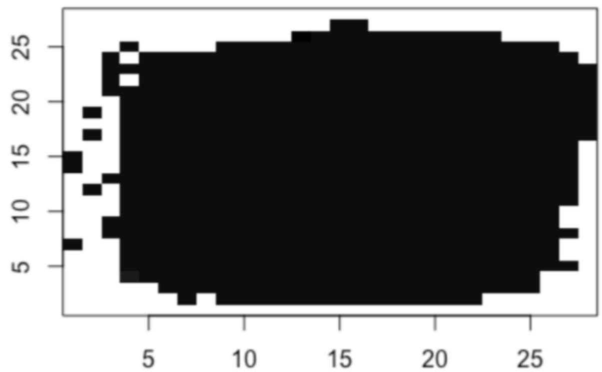
```
#The pair with the lowest error rate is 1 and 0
 df_label01 <- replace(train_Labels, which((train_Labels == 0) %in% TRUE), 1)
 df_label01 <- replace(df_label01, which((train_Labels == 1) %in% TRUE), -1)

df_y01 <- data.frame(df_label01)
df01 <- cbind(train_digits, df_y01)
df01 <- subset(df01,(df_label01==1|df_label01==-1))

set.seed(1)

sample <- sample(c(TRUE, FALSE), nrow(df01), replace=TRUE, prob=c(0.5,0.5))
train01    <- df01[sample, ]
test01  <- df01[!sample, ]
fit_train01 <- lm(train01$df_label01 ~ ., data = train01)
set.seed(1)
empty_list_beta01 <- vector("list", 784)
beta_image01 <- data.frame(fit_train01$coefficients)
i <- 2
for (num01 in not_zero_index){

  empty_list_beta01[num01] <- beta_image01[i,]
  i <- i+1

}
empty_matrix_beta01 <- matrix(empty_list_beta01, nrow = 28, ncol = 28)
empty_matrix_beta01 <- replace(empty_matrix_beta01, empty_matrix_beta01== 'NULL', NA)
image(1:28, 1:28, matrix(as.matrix(as.numeric(empty_matrix_beta01)), nrow=28)[ , 28:1],
      col = gray(seq(0, 1, 0.05)), xlab = "", ylab="")
```

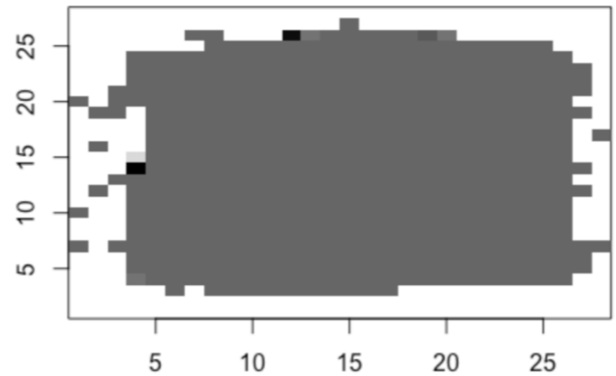
Beta Image for highest training set

```
#The pair with the lowest error rate is 1 and 9
 df_label91 <- replace(train_Labels, which((train_Labels == 1) %in% TRUE), 1)
 df_label91 <- replace(df_label91, which((train_Labels == 9) %in% TRUE), -1)

df_y91 <- data.frame(df_label91)
df91 <- cbind(train_digits, df_y91)
df91 <- subset(df91,(df_label91==1|df_label91==-1))

set.seed(1)

sample <- sample(c(TRUE, FALSE), nrow(df91), replace=TRUE, prob=c(0.5,0.5))
train91   <- df91[sample, ]
test91   <- df91[!sample, ]
fit_train91T <- lm(train91$df_label91 ~ ., data = train91)

set.seed(1)
empty_list_beta91 <- vector("list", 784)
beta_image91 <- data.frame(fit_train91T$coefficients)
i <- 2
for (num91 in not_zero_index){

  empty_list_beta91[num91] <- beta_image91[i,]
  i <- i+1

}
empty_matrix_beta91 <- matrix(empty_list_beta91, nrow = 28, ncol = 28)
empty_matrix_beta91 <- replace(empty_matrix_beta91, empty_matrix_beta91== 'NULL',
NA)
```

```
image(1:28, 1:28, matrix(as.matrix(as.numeric(empty_matrix_beta91)), nrow=28)[ ,
28:1],
      col = gray(seq(0, 1, 0.05)), xlab = "", ylab="")
```

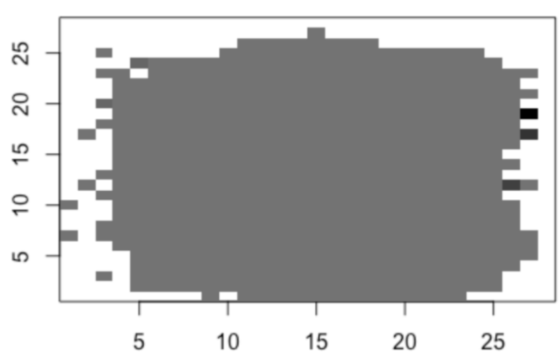
Beta Image for highest and lowest error rates for testing side by side

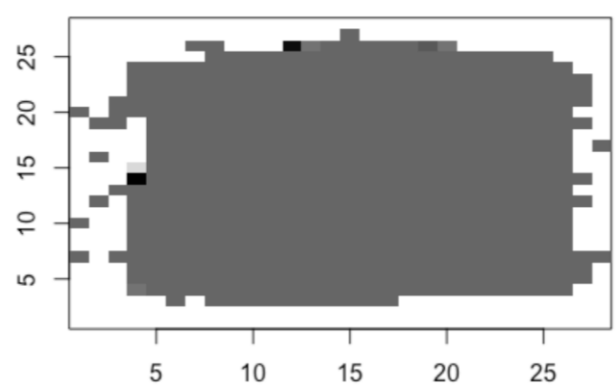




The coefficients for the highest error rate pair (5 and 8) are much higher then the lowest error rate pair(0 and 1).

Beta Image for highest and lowest error rates for training side by side





Part 3

Logistic Regression

Logistic Regression is usually used to predict the probability of a binary event occurring (yes or no)

We will fit the model using logistic regression. Our k will be 9. Similar to linear regression, the data will be split into train and test.

```
k = 9
df_label_log <- replace(train_Labels, which((train_Labels == k) %in% TRUE), 1)
df_label_log <- replace(df_label_log, which((train_Labels == k) %in% FALSE), -1)
df_y_log <- data.frame(df_label_log)
df_log <- cbind(train_digits, df_y_log)

set.seed(1)

sample <- sample(c(TRUE, FALSE), nrow(df_log), replace=TRUE, prob=c(0.5,0.5))
train_log  <- df_log[sample, ]
test_log   <- df_log[!sample, ]


log_fit <- glm(as.factor(df_label_log) ~ ., data = train_log, family = 'binomial')

log_fit
```

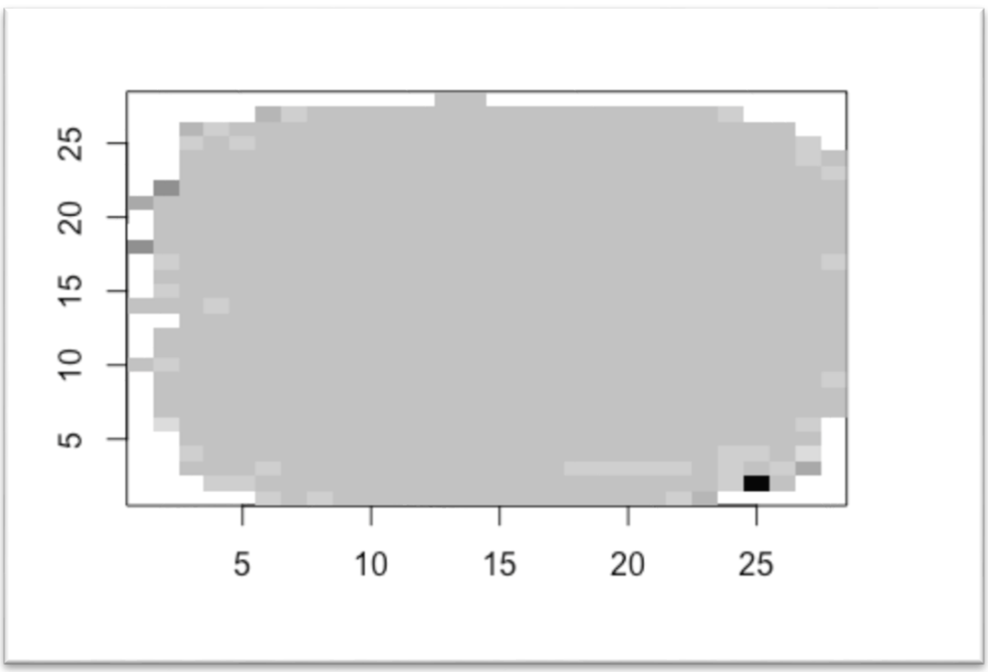
```
## Call:  glm(formula = as.factor(df_label_log) ~ ., family = "binomial",
##     data = train_log)
 ##
## Coefficients:
 ## (Intercept)          X13          X14          X15          X16          X33
##  -1.360e+15   -5.263e+13    1.493e+13           NA           NA           NA
##         X34          X35          X36          X37          X38          X39
##  -4.950e+14    1.752e+13    3.142e+12   -9.345e+12    3.913e+12    2.802e+12
##         X40          X41          X42          X43          X44          X45
##   5.330e+12   -9.864e+12    3.662e+12   -3.394e+10    3.984e+11    1.736e+12
##         X46          X47          X48          X49          X50          X51
##  -5.962e+12    5.804e+12   -6.944e+12   -1.612e+12   -3.662e+12    2.765e+13
##         X52          X59          X60          X61          X62          X63
##  -3.637e+13   -3.579e+13    1.355e+14   -4.025e+12   -3.094e+13    1.090e+13
##         X64          X65          X66          X67          X68          X69
##  -1.700e+13    1.122e+13   -4.306e+11   -1.342e+13    3.031e+12   -1.279e+12
##         X70          X71          X72          X73          X74          X75
##  -1.390e+12   -4.357e+11    3.224e+12   -7.568e+12    1.118e+12   -6.343e+12
##         X76          X77          X78          X79          X80          X81
##   5.430e+12    4.247e+12   -9.643e+12    3.417e+12    2.235e+12   -5.581e+12
##         X82          X87          X88          X89          X90          X91
##   1.957e+13    3.357e+13   -1.971e+13    1.482e+12    9.291e+12   -2.239e+12
##         X92          X93          X94          X95          X96          X97
##   5.568e+12   -6.724e+12   -2.258e+12    4.762e+12    3.894e+10    2.098e+11
##         X98          X99         X100         X101         X102         X103
##   9.761e+11   -5.728e+12    9.073e+11   -4.293e+12    4.762e+12   -5.118e+12
##        X104         X105         X106         X107         X108         X109
##  -4.776e+11    2.163e+12   -2.032e+12   -4.712e+10   -3.316e+12    1.433e+12
##        X110         X111         X114         X115         X116         X117
##  -7.187e+12    4.854e+13           NA   -2.051e+13    1.151e+13    5.103e+12
##        X118         X119         X120         X121         X122         X123
##   3.468e+12   -7.843e+11   -8.829e+11    1.813e+12    2.432e+12   -2.838e+11
##        X124         X125         X126         X127         X128         X129
##  -2.393e+12    8.565e+11   -4.324e+12    3.830e+12   -3.702e+12    2.627e+12
##        X130         X131         X132         X133         X134         X135
##  -2.341e+12    2.270e+12   -6.311e+12    4.366e+12    2.696e+12    3.551e+11
```

```
##        X136        X137        X138        X139        X140        X143
##   1.246e+12  -4.590e+11  -1.055e+13   9.852e+12  -8.159e+13   5.177e+12
##        X144        X145        X146        X147        X148        X149
##   2.386e+12  -5.368e+12   2.059e+12  -6.010e+12   1.598e+12  -3.847e+11
##        X150        X151        X152        X153        X154        X155
##   1.104e+11  -7.388e+11   1.938e+12  -1.312e+12   6.282e+11  -7.958e+11
##        X156        X157        X158        X159        X160        X161
##  -6.619e+11  -2.492e+11  -2.088e+12  -1.265e+12   4.210e+11  -3.533e+12
##        X162        X163        X164        X165        X166        X167
##   4.322e+11  -1.772e+12   1.735e+12  -8.310e+12   9.986e+12  -5.831e+12
##        X168        X170        X171        X172        X173        X174
##   3.498e+13  -1.205e+12   4.978e+12  -1.936e+12  -2.258e+12   2.911e+12
##        X175        X176        X177        X178        X179        X180
##   3.092e+11  -2.830e+12  -1.677e+12   1.143e+12  -1.760e+12   6.484e+11
##        X181        X182        X183        X184        X185        X186
##  -5.980e+10   1.081e+12   2.910e+11   4.937e+11   2.455e+11   9.034e+11
##        X187        X188        X189        X190        X191        X192
##   9.760e+11   2.484e+11  -2.809e+11   2.965e+11   2.401e+11  -1.944e+12
##        X193        X194        X195        X196        X197        X198
##   2.329e+12  -1.542e+12   3.407e+12  -6.669e+12   2.476e+15   2.340e+12
##        X199        X200        X201        X202        X203        X204
##  -3.233e+12   2.070e+12   8.428e+12  -7.379e+12   1.021e+12  -6.089e+11
##        X205        X206        X207        X208        X209        X210
##   1.663e+11  -4.657e+11   3.144e+11  -3.591e+11   1.045e+12   5.671e+11
##        X211        X212        X213        X214        X215        X216
##   9.616e+11   3.599e+11   1.591e+12  -1.511e+11   2.431e+10  -5.235e+11
##        X217        X218        X219        X220        X221        X222
##   3.301e+11  -4.316e+11  -1.430e+11   4.998e+11  -1.361e+12  -3.141e+12
##        X223        X224        X225        X226        X227        X228
##   1.482e+12  -1.151e+13          NA   7.896e+12  -1.782e+12  -2.226e+12
##        X229        X230        X231        X232        X233        X234
##  -6.501e+11   2.889e+11  -7.158e+09  -2.088e+12   4.751e+11  -5.298e+11
##        X235        X236        X237        X238        X239        X240
##   1.519e+11  -6.318e+11  -6.146e+11   3.966e+11   7.733e+11   7.602e+10
##        X241        X242        X243        X244        X245        X246
##   3.000e+11   1.436e+10   3.841e+11   6.214e+09  -7.662e+10   1.047e+12
##        X247        X248        X249        X250        X251        X252
##  -1.570e+12  -2.271e+11   2.515e+11  -1.064e+11   2.562e+12   8.457e+11
##        X253        X254        X255        X256        X257        X258
##  -3.859e+15  -1.077e+13   4.371e+12   2.066e+12  -2.485e+12  -4.683e+11
##        X259        X260        X261        X262        X263        X264
##  -1.827e+12   3.818e+11  -3.146e+11  -3.795e+11   1.902e+09   1.928e+11
##        X265        X266        X267        X268        X269        X270
##  -3.701e+11  -4.273e+11   3.183e+11   7.664e+11  -2.006e+11  -7.962e+11
##        X271        X272        X273        X274        X275        X276
##   1.550e+11  -9.894e+11  -1.252e+11  -7.767e+11   3.789e+11  -2.549e+12
##        X277        X278        X279        X280        X281        X282
##   1.766e+11  -4.004e+12   2.912e+12   1.739e+12   3.332e+15  -1.444e+12
##        X283        X284        X285        X286        X287        X288
##   2.043e+12  -1.922e+12  -1.373e+12   9.407e+11   1.861e+12  -9.762e+11
##        X289        X290        X291        X292        X293        X294
##   4.364e+11   5.565e+11  -4.385e+11  -8.811e+10   1.652e+10  -1.034e+12
##        X295        X296        X297        X298        X299        X300
##   5.709e+11  -1.097e+12  -1.756e+11   2.895e+11  -3.355e+11   2.836e+11
##        X301        X302        X303        X304        X305        X306
##  -4.779e+09  -8.317e+11   3.899e+11   2.128e+12  -8.608e+11  -5.481e+12
##        X307        X308        X309        X310        X311        X312
##   4.736e+12  -1.797e+13          NA  -3.291e+11  -1.339e+13   8.990e+11
##        X313        X314        X315        X316        X317        X318
##   6.744e+11  -5.519e+11  -1.957e+11   1.037e+12   6.358e+11  -5.494e+11
##        X319        X320        X321        X322        X323        X324
##   9.056e+11   6.507e+11   8.063e+11  -9.214e+11  -8.340e+11   8.543e+11
##        X325        X326        X327        X328        X329        X330
##  -3.330e+11   4.466e+11   5.705e+11   1.600e+11   1.029e+12   1.117e+12
##        X331        X332        X333        X334        X335        X336
##   1.706e+11  -1.944e+12   1.207e+12  -1.518e+12  -1.045e+12   4.074e+13
##        X337        X338        X339        X340        X341        X342
##          NA  -9.089e+12   2.112e+12   1.812e+12  -1.486e+12   9.025e+11
##        X343        X344        X345        X346        X347        X348
##  -1.216e+11  -7.783e+11   1.036e+11   7.901e+11  -1.190e+11   2.534e+10
##        X349        X350        X351        X352        X353        X354
```

```
##   3.272e+11   -3.311e+11   -4.921e+10    8.973e+11    4.763e+11    3.520e+11
##        X355         X356         X357         X358         X359         X360
##   2.317e+11    1.532e+11    3.581e+11    1.436e+11    4.546e+11    3.236e+12
##        X361         X362         X363         X364         X365         X366
##  -2.641e+12   -5.697e+12   -7.509e+12   -1.589e+13           NA    4.926e+11
##        X367         X368         X369         X370         X371         X372
##  -7.029e+12   -5.485e+12    2.504e+12    9.537e+11    2.411e+11    1.006e+12
##        X373         X374         X375         X376         X377         X378
##   2.675e+11   -1.124e+11    3.095e+11   -1.613e+11    1.044e+11   -6.309e+11
##        X379         X380         X381         X382         X383         X384
##   9.070e+11   -1.769e+11    1.007e+12   -4.363e+11    6.833e+11    9.295e+11
##        X385         X386         X387         X388         X389         X390
##  -2.195e+11    1.413e+12   -7.292e+11   -1.986e+12    5.987e+12   -7.614e+12
##        X391         X392         X393         X394         X395         X396
##   6.445e+12   -5.757e+12    5.541e+12   -1.915e+13    1.347e+13    6.158e+12
##        X397         X398         X399         X400         X401         X402
##  -3.059e+12   -1.091e+12    7.576e+11   -4.199e+11   -3.009e+11    2.100e+11
##        X403         X404         X405         X406         X407         X408
##   6.620e+10    2.448e+11    1.356e+12   -9.536e+11   -5.539e+10    4.715e+10
##        X409         X410         X411         X412         X413         X414
##   7.029e+11    6.102e+11   -1.149e+11    3.520e+10   -2.176e+11   -7.626e+11
##        X415         X416         X417         X418         X419         X420
##   1.375e+12   -1.484e+12   -4.475e+12    8.440e+12   -7.120e+12    2.580e+13
##        X421         X422         X423         X424         X425         X426
##          NA    1.353e+15   -5.833e+12   -5.158e+12    1.915e+12    2.332e+11
##        X427         X428         X429         X430         X431         X432
##  -2.666e+11   -2.767e+11    6.126e+11    4.481e+10   -7.782e+11    8.115e+11
##        X433         X434         X435         X436         X437         X438
##   1.702e+11   -8.742e+11    2.546e+11   -4.766e+11    1.248e+12   -2.174e+11
##        X439         X440         X441         X442         X443         X444
##  -4.021e+11   -4.033e+09    3.936e+11    1.251e+12   -2.097e+12    1.574e+12
##        X445         X446         X447         X448         X449         X450
##  -1.144e+12   -2.876e+12    6.561e+12   -3.600e+12           NA    3.495e+12
##        X451         X452         X453         X454         X455         X456
##  -1.272e+11   -4.839e+11    2.562e+12    6.466e+11   -9.737e+11    2.716e+11
##        X457         X458         X459         X460         X461         X462
##  -4.032e+11    2.129e+11    3.206e+09    1.689e+12   -4.237e+11   -8.629e+11
##        X463         X464         X465         X466         X467         X468
##  -2.298e+10    2.753e+11    2.060e+11    3.392e+11   -2.733e+10    5.864e+11
##        X469         X470         X471         X472         X473         X474
##  -8.590e+11   -2.603e+12    3.407e+12   -3.336e+12    2.042e+12   -1.839e+12
##        X475         X476         X478         X479         X480         X481
##  -1.132e+13    1.152e+13    4.021e+13   -9.732e+11    2.698e+12   -4.185e+12
##        X482         X483         X484         X485         X486         X487
##  -1.443e+12    6.970e+11   -7.196e+11    1.781e+12   -1.243e+12    6.787e+11
##        X488         X489         X490         X491         X492         X493
##  -5.773e+11    7.576e+11   -1.854e+11   -6.397e+11   -7.003e+11   -8.674e+11
##        X494         X495         X496         X497         X498         X499
##   4.042e+11   -1.143e+11   -8.216e+11    1.035e+12   -1.340e+12   -3.312e+11
##        X500         X501         X502         X503         X504         X505
##   1.158e+12   -3.957e+12   -9.068e+11    8.685e+12   -2.111e+13    2.924e+13
##        X506         X507         X508         X509         X510         X511
##  -2.184e+13   -8.921e+12   -1.190e+12    3.107e+12   -6.188e+11   -6.358e+10
##        X512         X513         X514         X515         X516         X517
##   4.450e+11   -7.358e+11    5.793e+11   -3.510e+11    1.340e+12   -1.369e+12
##        X518         X519         X520         X521         X522         X523
##   1.214e+10    8.295e+11    1.315e+12    1.793e+11   -2.858e+10    2.518e+09
##        X524         X525         X526         X527         X528         X529
##  -5.487e+11   -3.877e+11    1.039e+12   -1.744e+12    5.541e+11    1.944e+12
##        X530         X531         X532         X533         X534         X535
##   1.286e+12   -6.021e+12   -4.689e+12           NA   -6.763e+12    3.060e+12
##        X536         X537         X538         X539         X540         X541
##   2.077e+12   -1.624e+12    4.046e+11   -2.202e+12    8.475e+11    1.159e+12
##        X542         X543         X544         X545         X546         X547
##  -1.088e+12   -3.033e+11   -1.425e+12    8.132e+11   -1.126e+11   -1.032e+12
##        X548         X549         X550         X551         X552         X553
##  -1.358e+12   -4.077e+11   -4.330e+11   -6.225e+11   -2.174e+11   -1.641e+12
##        X554         X555         X556         X557         X558         X559
##   6.168e+11   -3.373e+11    1.536e+12   -1.584e+12   -6.257e+12    8.736e+12
##        X560         X562         X563         X564         X565         X566
##   3.179e+13   -5.095e+12    3.656e+12   -5.410e+12   -2.683e+11    8.888e+10
```

```
##        X567         X568         X569         X570         X571         X572
##   3.188e+12   -4.525e+12   -6.679e+11   -1.327e+12    2.491e+11   -9.263e+10
##        X573         X574         X575         X576         X577         X578
##  -9.344e+11   -1.652e+11    7.397e+10    6.209e+11   -3.737e+11    1.918e+11
##        X579         X580         X581         X582         X583         X584
##   7.608e+11    6.712e+11    1.273e+12    1.134e+10    8.571e+11   -3.039e+12
##        X585         X586         X587         X588         X589         X590
##   1.476e+12    7.645e+12   -1.119e+13    2.262e+13           NA    9.401e+12
##        X591         X592         X593         X594         X595         X596
##  -4.758e+12   -1.893e+11    2.642e+12   -1.581e+12    1.292e+12    4.895e+11
##        X597         X598         X599         X600         X601         X602
##   9.601e+11   -6.076e+10   -6.636e+11   -5.975e+11   -9.747e+11    6.173e+11
##        X603         X604         X605         X606         X607         X608
##  -9.007e+11   -3.027e+11    5.920e+10   -1.293e+12   -9.957e+11   -4.944e+11
##        X609         X610         X611         X612         X613         X614
##  -1.361e+11   -1.078e+12   -1.154e+12    2.150e+12    1.483e+12   -5.556e+12
##        X615         X616         X617         X618         X619         X620
##  -1.189e+13    7.447e+13           NA    8.038e+13    3.166e+12   -3.050e+11
##        X621         X622         X623         X624         X625         X626
##   2.656e+12   -1.781e+12   -4.788e+10   -1.848e+12    2.363e+09   -3.307e+11
##        X627         X628         X629         X630         X631         X632
##   4.840e+11    5.208e+11    9.518e+10   -5.775e+11   -3.163e+11    3.479e+10
##        X633         X634         X635         X636         X637         X638
##   8.271e+10    9.082e+11    7.996e+11   -8.462e+11    3.016e+11    9.110e+11
##        X639         X640         X641         X642         X643         X644
##  -7.421e+11   -2.604e+12    4.640e+12    1.135e+11    2.045e+13           NA
##        X647         X648         X649         X650         X651         X652
##  -4.965e+12   -1.745e+10    3.146e+11   -2.913e+12    3.088e+11   -1.302e+12
##        X653         X654         X655         X656         X657         X658
##   1.310e+11   -9.891e+11   -1.282e+12   -3.815e+11   -3.792e+11   -6.084e+10
##        X659         X660         X661         X662         X663         X664
##  -9.762e+11   -1.739e+12   -9.658e+11   -2.177e+12   -1.282e+12    8.741e+10
##        X665         X666         X667         X668         X669         X670
##  -1.306e+12   -1.392e+12    6.545e+12   -2.165e+12    2.159e+12   -1.665e+13
##        X671         X675         X676         X677         X678         X679
##  -1.516e+13    1.561e+13   -2.309e+13    2.902e+12    1.714e+12   -1.289e+11
##        X680         X681         X682         X683         X684         X685
##   5.932e+11    7.407e+11    7.170e+11   -4.943e+11   -4.763e+11   -6.012e+11
##        X686         X687         X688         X689         X690         X691
##  -5.448e+11    8.049e+11    9.546e+11    6.449e+11    1.077e+12    4.098e+11
##        X692         X693         X694         X695         X696         X697
##   1.432e+12    9.725e+11    5.993e+11   -5.614e+12    1.028e+13   -1.364e+13
##        X698         X699         X703         X704         X705         X706
##   1.275e+13    1.896e+14    1.127e+14    8.603e+12   -9.369e+12    3.042e+12
##        X707         X708         X709         X710         X711         X712
##  -1.609e+12   -1.229e+12    4.856e+11   -2.947e+11    1.500e+11    5.763e+10
##        X713         X714         X715         X716         X717         X718
##   1.360e+12   -6.243e+11   -2.240e+11   -4.232e+11    1.932e+11    7.432e+11
##        X719         X720         X721         X722         X723         X724
##  -8.984e+11    1.088e+12    2.384e+12   -7.992e+11    4.038e+12   -7.819e+12
##        X725         X726         X727         X732         X733         X734
##   9.899e+12    4.667e+13   -2.349e+14    8.224e+13    5.988e+12    5.467e+11
##        X735         X736         X737         X738         X739         X740
##   9.770e+11    1.446e+12    5.061e+10    1.369e+12    7.586e+11    3.359e+11
##        X741         X742         X743         X744         X745         X746
##   2.501e+11    4.901e+11    1.649e+12    7.420e+11   -5.299e+11   -2.538e+10
##        X747         X748         X749         X750         X751         X752
##   2.700e+12   -3.128e+12    2.210e+12    2.698e+11    1.551e+12    2.008e+13
##        X753         X754         X761         X762         X763         X764
##  -6.327e+14    3.533e+13           NA    3.724e+13   -7.722e+12    6.415e+11
##        X765         X766         X767         X768         X769         X770
##   1.420e+12   -1.640e+12    2.556e+11   -5.791e+12    2.896e+12   -1.051e+13
##        X771         X772         X773         X774         X775         X776
##   5.186e+12   -1.166e+13    8.552e+12   -7.442e+12    1.200e+12    8.273e+12
##        X777         X778         X779         X780
##  -1.265e+13    3.927e+13   -9.350e+13           NA
##
## Degrees of Freedom: 30048 Total (i.e. Null);  29347 Residual
 ## Null Deviance:       19460
## Residual Deviance: 83190     AIC: 84590
```

```r
empty_list_beta_log <- vector("list", 784)
beta_image_log <- data.frame(log_fit$coefficients)
i_log <- 2
for (num_log in not_zero_index){

  empty_list_beta_log[num_log] <- beta_image[i_log,]
  i_log <- i_log+1

}
empty_matrix_beta_log <- matrix(empty_list_beta, nrow = 28, ncol = 28)
empty_matrix_beta_log <- replace(empty_matrix_beta_log, empty_matrix_beta_log==
'NULL', NA)
image(1:28, 1:28, matrix(as.matrix(as.numeric(empty_matrix_beta_log)), nrow=28)[ ,
28:1],
      col = gray(seq(0, 1, 0.05)), xlab = "", ylab="")
```




```r
predicted_log <- predict(log_fit, test_log, type="response")
Confusion_Matrix_log <- ifelse(predicted_log >.5,"1","-1")

confusion_mat_test_log <- table(Confusion_Matrix_log, test_log[['df_label_log']])
confusion_mat_test_log
```

```
Confusion_Matrix_log     -1      1
                   -1 26328    719
                   1    661   2243
```

```
predicted_log <- predict(log_fit, train_log, type="response")
Confusion_Matrix_log <- ifelse(predicted_log >.5,"1","-1")

confusion_mat_train_log <- table(Confusion_Matrix_log, train_log[['df_label_log']])
confusion_mat_train_log
```

```
Confusion_Matrix_log     -1      1
                   -1 26562    654
                   1    500   2333
```

```
TP <- confusion_mat_test_log[1,1]
TN <- confusion_mat_test_log[2,2]
FP <- confusion_mat_test_log[1,2]
FN <- confusion_mat_test_log[2,1]
Accuracy_test <- (TP + TN) / (TP + FP + TN + FN)
classification_error_test_log <- 1-Accuracy_test
paste0('The Accuracy for the testing set is: ',Accuracy_test)
paste0('The classification error rate for the testing set is: 
',classification_error_test_log)
```

"The Accuracy for the testing set is: **0.953924743748122**"
"The classification error rate for the testing set is: **0.046075256251878**"

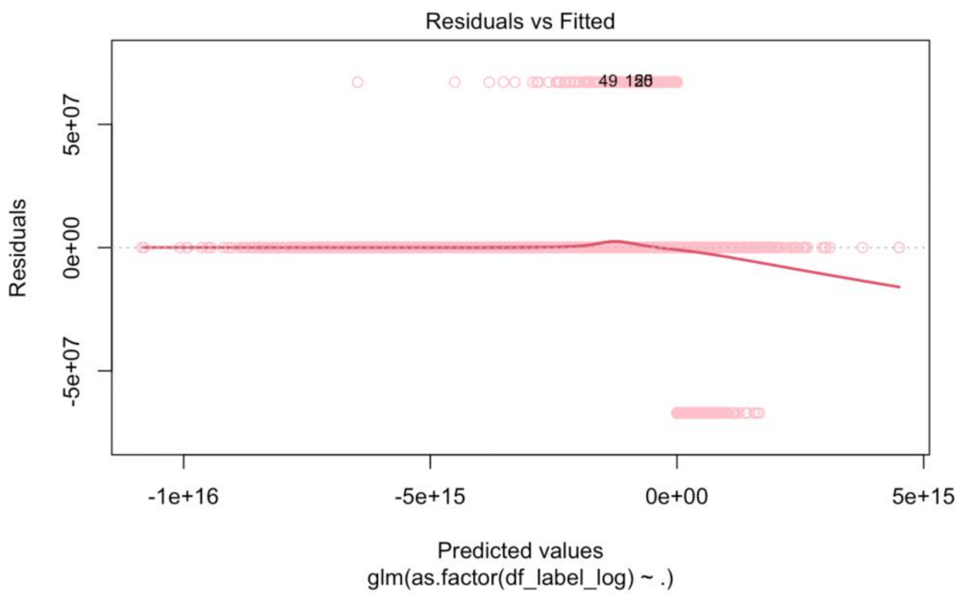
```
TP <- confusion_mat_train_log[1,1]
TN <- confusion_mat_train_log[2,2]
FP <- confusion_mat_train_log[1,2]
FN <- confusion_mat_train_log[2,1]
Accuracy <- (TP + TN) / (TP + FP + TN + FN)
classification_error_train_log <- 1-Accuracy
paste0('The Accuracy for the training set is: ',Accuracy_test)
paste0('The classification error rate for the training set is: 
',classification_error_train_log)
```

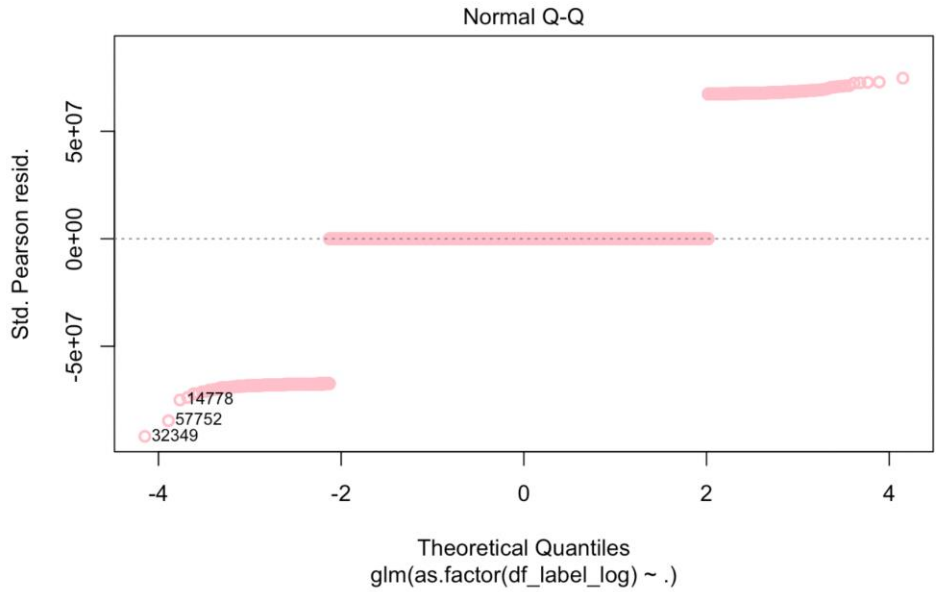
"The Accuracy for the training set is: **0.953924743748122**"

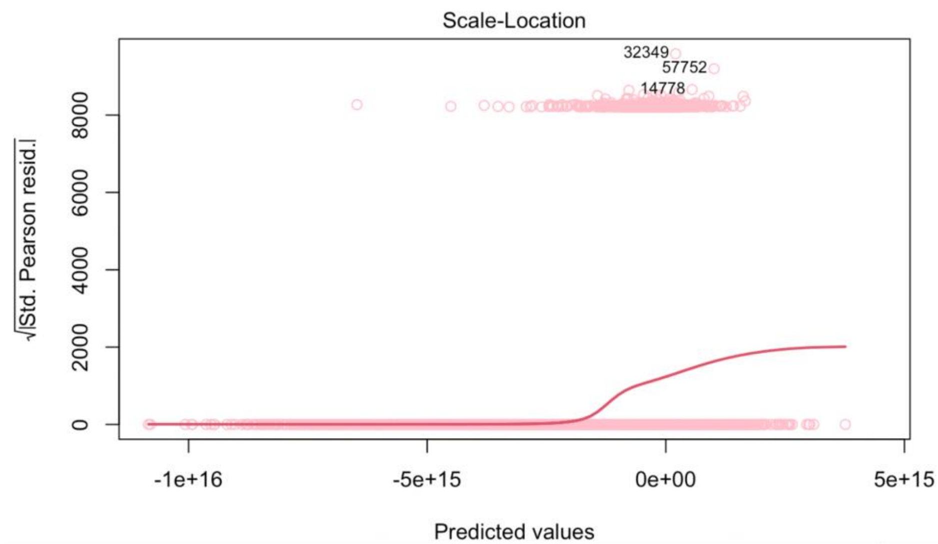
```
"The classification error rate for the training set is:
0.0384039402309561"
```

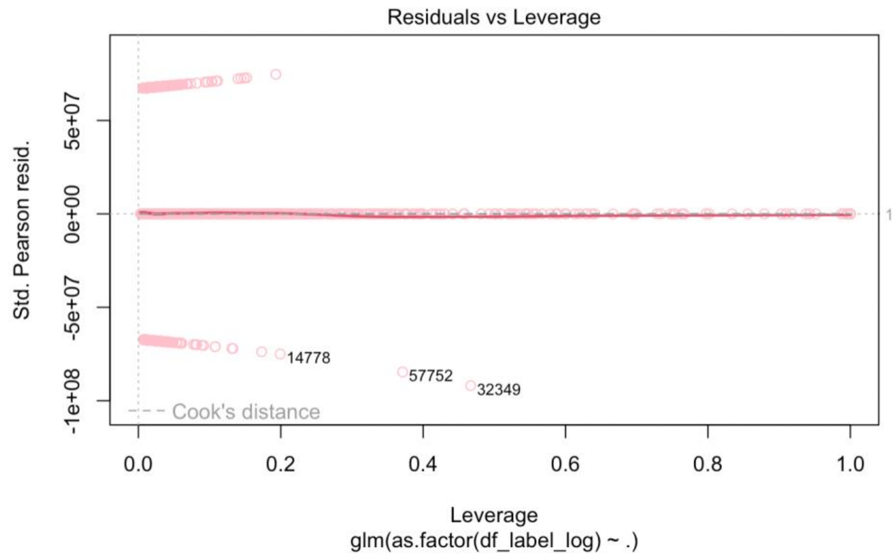
The Classification Error Rates for train and test dataset are similar but it is a little higher for the testing set: the testing error rate is 0.04607 and the training error rate is 0.0384. Compared to our Linear Regression Model, the error rates for Logistics Regression are lower for both the training and testing set.

```
plot(log_fit, lwd = '2', col = 'pink')
```




Normal Q-Q
Std. Pearson resid.
5e+07
0e+00
-5e+07
14778
57752
32349
-4
-2
0
2
4
Theoretical Quantiles
glm(as.factor(df_label_log) ~ .)


Scale-Location
32349
57752
14778
√|Std. Pearson resid.|
8000
6000
4000
2000
0
-1e+16
-5e+15
0e+00
5e+15
Predicted values



The QQ plot indicates that there might be extreme values that would not be expected in a normal distribution. The residual for both Residual vs Fitted and Residual vs Leverage are very high. The dataset should be reevaluated for outliers.

Part 2 with Logistic Regression

Logistic regression is applied on all pairs on digit, similar to part 2. These 2 chunks of code are similar but with different variable names.

```
label = list()
 list_df_log = list()
 list_of_pairs_log = list()
 for (x_log in 0:9){
   corr_log = x_log
   list_to_nine_log <- list(0,1,2,3,4,5,6,7,8,9)
   second_loop_list_log <- list_to_nine_log[-c(corr_log+1)]

  for (y_log in 0:9) {
     if (y_log > x_log) {
       incorr_log = y_log
       pair_log = list()
       pair_log <- append(pair_log,corr_log)
       pair_log <- append(pair_log,incorr_log)
       list_of_pairs_log[[length(list_of_pairs_log) + 1]] <- pair_log
       df_label2_log <- replace(train_Labels, which((train_Labels == corr_log) %in%
TRUE), 1)
       df_label2_log <- replace(df_label2_log, which((train_Labels == incorr_log)
%in% TRUE), -1)
       list_df_log[[length(list_df_log) + 1]] <- df_label2_log
     }
   }
}
```

The dataframes for each of the digit pairs are stored in list_of_df_lin_log.

```
list_of_df_lin_log = list()
 for (one_pair_log in list_df_log){
   df_y2_log <- data.frame(one_pair_log)
   df2_log <- cbind(train_digits, df_y2_log)
   df2_log <- subset(df2_log,(one_pair_log==1|one_pair_log==-1))
   list_of_df_lin_log[[length(list_of_df_lin_log) + 1]] <- df2_log

}
```

The Logistic Regression is applied to all dataframes from above. The confusion matrix, accuracy and classification error rates are calculated for the train and test sets.

```
set.seed(1)
 train_error_log <- list()
 test_error_log <- list()
 for (df_for_split_log in list_of_df_lin_log){
   sample_log <- sample(c(TRUE, FALSE), nrow(df_for_split_log), replace=TRUE,
prob=c(0.5,0.5))
   train01_log  <- df_for_split_log[sample_log, ]
   test01_log   <- df_for_split_log[!sample_log, ]
   fit_train01_log <- glm(as.factor(one_pair_log) ~ ., data = train01_log, family =
'binomial')

  predicted01_log <- predict(fit_train01_log, test01_log, type="response")
  p_class01_log <- ifelse(predicted01_log >.5,"1","-1")
  confusion_mat_test_log1 <- table(p_class01_log, test01_log[['one_pair_log']])



  predicted_log1 <- predict(fit_train01_log, train01_log, type="response")
  p_class_log1 <- ifelse(predicted_log1 >.5,"1","-1")
  confusion_mat_train_log1 <- table(p_class_log1, train01_log[['one_pair_log']])



  TP <- confusion_mat_test_log1[1,1]
  TN <- confusion_mat_test_log1[2,2]
  FP <- confusion_mat_test_log1[1,2]
  FN <- confusion_mat_test_log1[2,1]
  Accuracy_test <- (TP + TN) / (TP + FP + TN + FN)
  classification_error_test_log1 <- 1-Accuracy_test
  test_error_log[[length(test_error_log) + 1]] <- classification_error_test_log1

  TP <- confusion_mat_train_log1[1,1]
  TN <- confusion_mat_train_log1[2,2]
  FP <- confusion_mat_train_log1[1,2]
  FN <- confusion_mat_train_log1[2,1]
  Accuracy <- (TP + TN) / (TP + FP + TN + FN)
  classification_error_train_log1 <- 1-Accuracy
  train_error_log[[length(train_error_log) + 1]] <- classification_error_train_log1



}
```

Similar to part 2, the error rates for testing are placed on the lower matrix while the error rates for training are placed on the upper matrix A 10x10 matrix is created and lower.tri and upper.tri are used to fill in the matrix. The diagonal does not have error rates.

```
suppressWarnings({

my_mat_log <- matrix(, ncol = 10, nrow=10)

my_mat_log[lower.tri(my_mat_log, diag = FALSE)] <- test_error_log[1:45]
my_mat_log <- matrix(my_mat_log, ncol = 10, nrow=10)
 my_mat_log[upper.tri(my_mat_log, diag = FALSE)] <- train_error_log[1:45]
my_mat_log_1 <- rbind(c(0:9), my_mat_log)
 my_mat_log_2 <- cbind(c(-1:9), my_mat_log_1)
 my_mat_log_2

})
```

-1	0	1	2	3	4	5	6	7	8	9
0	NA	0	0.01288437	0	0	0	0	0	0.03542673	0
1	0.005952381	NA	0.00921466	0.01423171	0.02784097	0	0	0.0271851	0	0.05373467
2	0.04932353	0.03722903	NA	0	0	0	0.543945	0	0	0
3	0.03958469	0.02911392	0.5552474	NA	0	0	0	0	0	0
4	0.01841761	0.01505829	0.03035178	0.021003	NA	0	0	0.01944173	0	0.5237736
5	0.04520458	0.02155242	0.04369469	0.04952129	0.03357048	NA	0	0	0.01725275	0
6	0.03159811	0.01485069	0.03007599	0.01837187	0.03084184	0.03601718	NA	0	0.02017478	0.01219
7	0.02334918	0.0259344	0.03526771	0.03123042	0.03681964	0.0285526	0.02394958	NA	0	0.02559492
8	0.05137318	0.044171	0.06449882	0.0605803	0.05109802	0.08215391	0.5195972	0.03874973	NA	0.003753351
9	0.02608789	0.01935381	0.0300655	0.03523859	0.04576857	0.0430854	0.02082884	0.05462583	0.03178909	NA

The results from this matrix are different than what I was expecting and what I got with the Linear Regression Model. I was expecting the results to be 1 and 7 or 5 and 8. The **highest** error rate is **0.55524** for the digit pair **2** and **3.** The top half of both these digits are very similar which could be an explanation as to why it has a high error rate. The lowest error rate is **0.00595** from the pair **0** and **1,** similar to the linear regression error rate.

Another thing to notice in this matrix is that there are a lot of 0s. The zeros are mostly on training set error rates. It is unlikely that a lot of the pairs have 100% accuracy. The zeros could be due to overfitting, which is only the training set (the dataset which we fitted) has the 0 error rates

Beta Image for Highest Error Rate

```
df_label_log32 <- replace(train_Labels, which((train_Labels == 2) %in% TRUE), 1)
df_label_log32 <- replace(df_label_log32, which((train_Labels == 3) %in% TRUE), -1)
df_y_log32 <- data.frame(df_label_log32)
df_log32 <- cbind(train_digits, df_y_log32)
```

```
set.seed(1)
sample <- sample(c(TRUE, FALSE), nrow(df_log32), replace=TRUE, prob=c(0.5,0.5))
train_log32  <- df_log32[sample, ]
test_log32   <- df_log32[!sample, ]
 #head(train_log, 10)
log_fit32 <- glm(as.factor(df_label_log32) ~ ., data = train_log32, family =
'binomial')
```

```
set.seed(1)
 empty_list_beta32 <- vector("list", 784)
 beta_image32 <- data.frame(log_fit32$coefficients)
 i <- 2
 for (num32 in not_zero_index){

  empty_list_beta32[num32] <- beta_image32[i,]
   i <- i+1


}


empty_matrix_beta32 <- matrix(empty_list_beta32, nrow = 28, ncol = 28)
 empty_matrix_beta32 <- replace(empty_matrix_beta32, empty_matrix_beta32== 'NULL',
NA)
image(1:28, 1:28, matrix(as.matrix(as.numeric(empty_matrix_beta32)), nrow=28)[ ,
28:1],
      col = gray(seq(0, 1, 0.05)), xlab = "", ylab="")
```

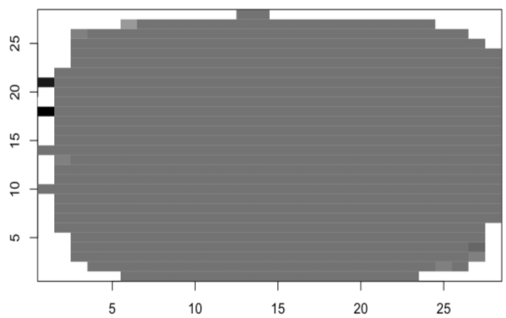
Beta Image for Lowest Error Rate

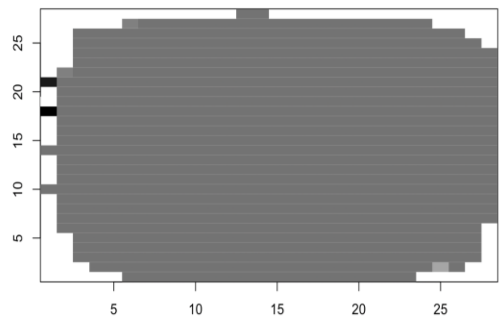
```
df_label_log01 <- replace(train_Labels, which((train_Labels == 1) %in% TRUE), 1)
df_label_log01 <- replace(df_label_log01, which((train_Labels == 0) %in% TRUE), -1)
df_y_log01 <- data.frame(df_label_log01)
df_log01 <- cbind(train_digits, df_y_log01)
set.seed(1)

sample <- sample(c(TRUE, FALSE), nrow(df_log01), replace=TRUE, prob=c(0.5,0.5))
train_log01  <- df_log01[sample, ]
test_log01   <- df_log01[!sample, ]
log_fit01 <- glm(as.factor(df_label_log01) ~ ., data = train_log01, family =
'binomial')
```

```
set.seed(1)
 empty_list_beta01 <- vector("list", 784)
 beta_image01 <- data.frame(log_fit01$coefficients)
 i <- 2
 for (num01 in not_zero_index){

  empty_list_beta01[num01] <- beta_image01[i,]
   i <- i+1


}
```

```
empty_matrix_beta01_log <- matrix(empty_list_beta01, nrow = 28, ncol = 28)
 empty_matrix_beta01_log <- replace(empty_matrix_beta01_log,
empty_matrix_beta01_log== 'NULL', NA)
image(1:28, 1:28, matrix(as.matrix(as.numeric(empty_matrix_beta01_log)), nrow=28)[ ,
28:1],
      col = gray(seq(0, 1, 0.05)), xlab = "", ylab="")
```

Beta image comparison for highest and lowest error rates.





The coefficients for both seem to be very similar.

Part 4

Removing outliers with cooks distance (Linear Regression on K vs not K)

Cooks distance will be used to find and remove outliers. The goal for this part is to see if the outliers impacted the results and whether removing them gives us better or worse results. For part 1, I used k = 9. I will repeat part 1 again for digit 9 but with outliers removed. This will be done with Linear and Logistic Regression.

Part 2 will be repeated as well for all pairs of digits and the error rate matrices will be compared

I'm creating a dataframe called df_cook which is a combination of the train_digits dataframe and the labels columns.

```
df_label_cook <- replace(train_Labels, which((train_Labels == 9) %in% TRUE), 1)
df_label_cook <- replace(df_label_cook, which((train_Labels == 9) %in% FALSE), -1)
dfY_cook <- data.frame(df_label_cook)
df_cook <- cbind(train_digits, dfY_cook)
```

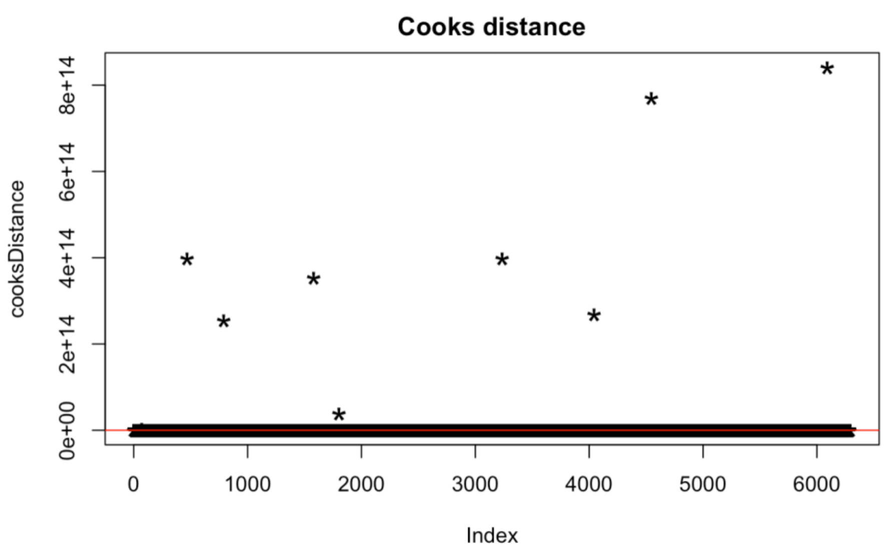
The df_cook is split into train and test and the train_cook is is used to fit the model using linear regression

```
set.seed(1)
sample <- sample(c(TRUE, FALSE), nrow(df_cook), replace=TRUE, prob=c(0.5,0.5))
```

```
train_cook  <- df_cook[sample, ]
test_cook   <- df_cook[!sample, ]
fit_cook <- lm(train_cook$df_label_cook ~ ., data = train_cook)
```

In order to get the cooks distance outliers, cooks.distance() is used on the fit model fit_cook. Outliers are numbers that are greater than (4/n), n being the number of rows. After the outliers are identified they are removed from the train_cook dataset.

```
cooksDistance <- cooks.distance(fit_cook)

sample_size <- nrow(fit_cook)
plot(cooksDistance, pch="*", cex=2, main="Cooks distance")
abline(h = 4/sample_size, col="red")

n <- nrow(train_cook)
outliers <- as.numeric(names(cooksDistance)[(cooksDistance > (4/n))])
index <- data.frame(outliers)
train_cook_removed <- train_cook[!(row.names(train_cook) %in% index$outliers),]
```



train_cook_removed is the train split with the outliers removed. Now the train split is fit. We will repeat the steps in part 1.

```
fit_outliers <- lm(train_cook_removed$df_label_cook ~ ., data = train_cook_removed)
predicted_cook <- predict(fit_outliers, test_cook, type="response")
p_class_cook <- ifelse(predicted_cook >.5,"1","-1")
confusion_mat_test_cook <- table(p_class_cook, test_cook[['df_label_cook']])
confusion_mat_test_cook
```

```
p_class_cook     -1      1
           -1 26864   2617
            1    125    345
```

```
predicted_cook <- predict(fit_outliers, train_cook, type="response")
p_class_cook <- ifelse(predicted_cook >.5,"1","-1")
confusion_mat_train_cook <- table(p_class_cook, train_cook[['df_label_cook']])
confusion_mat_train_cook
```

```
p_class_cook     -1      1
           -1 26946   2617
            1    116    370
```

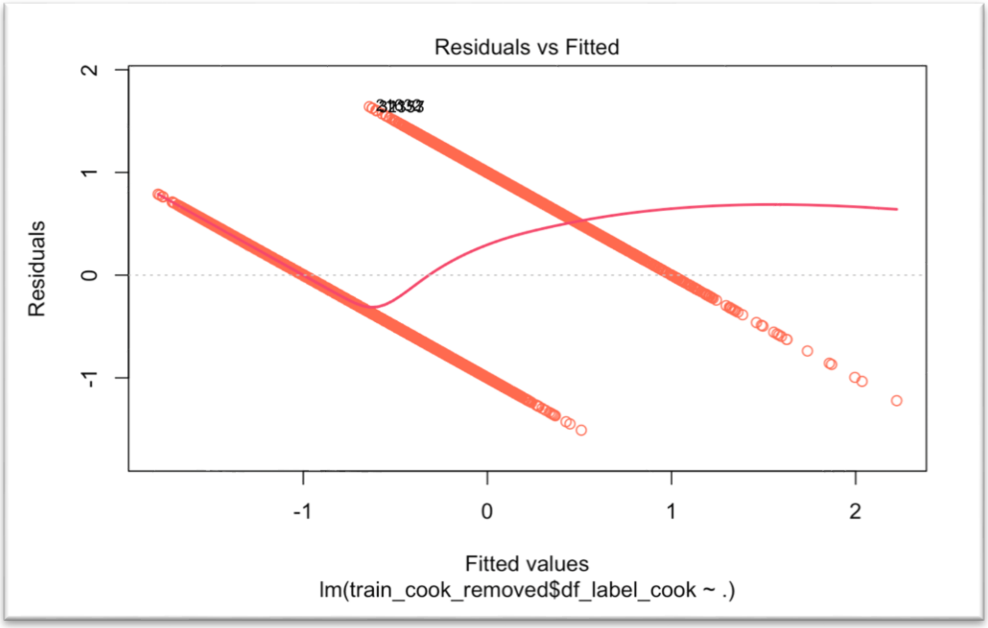
```
TP <- confusion_mat_test_cook[1,1]
TN <- confusion_mat_test_cook[2,2]
FP <- confusion_mat_test_cook[1,2]
FN <- confusion_mat_test_cook[2,1]
Accuracy_test <- (TP + TN) / (TP + FP + TN + FN)
classification_error_test <- 1-Accuracy_test
paste0('The classification error rate for the testing set is:
',classification_error_test)
```

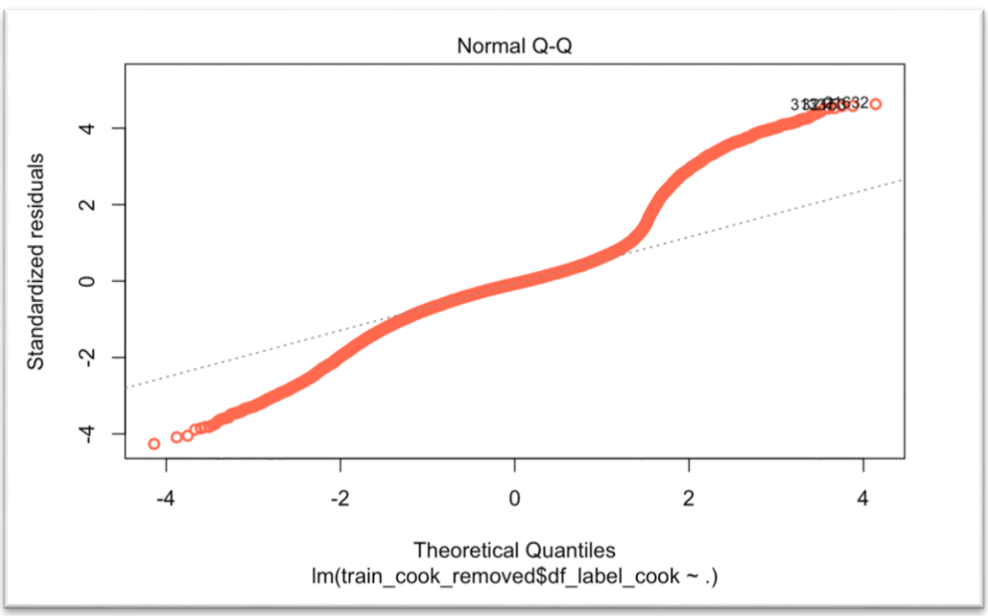
```
"The classification error rate for the testing set is:
0.0915495309004708"
```

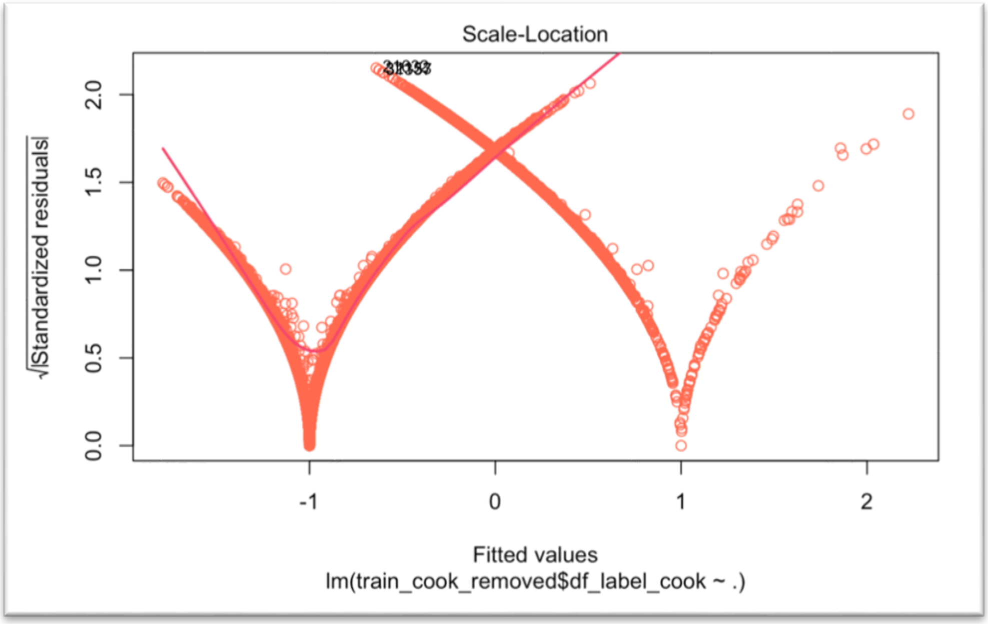
```
TP <- confusion_mat_train_cook[1,1]
TN <- confusion_mat_train_cook[2,2]
FP <- confusion_mat_train_cook[1,2]
FN <- confusion_mat_train_cook[2,1]
Accuracy <- (TP + TN) / (TP + FP + TN + FN)
classification_error_train <- 1-Accuracy
paste0('The classification error rate for the training set
is:',classification_error_train)
```

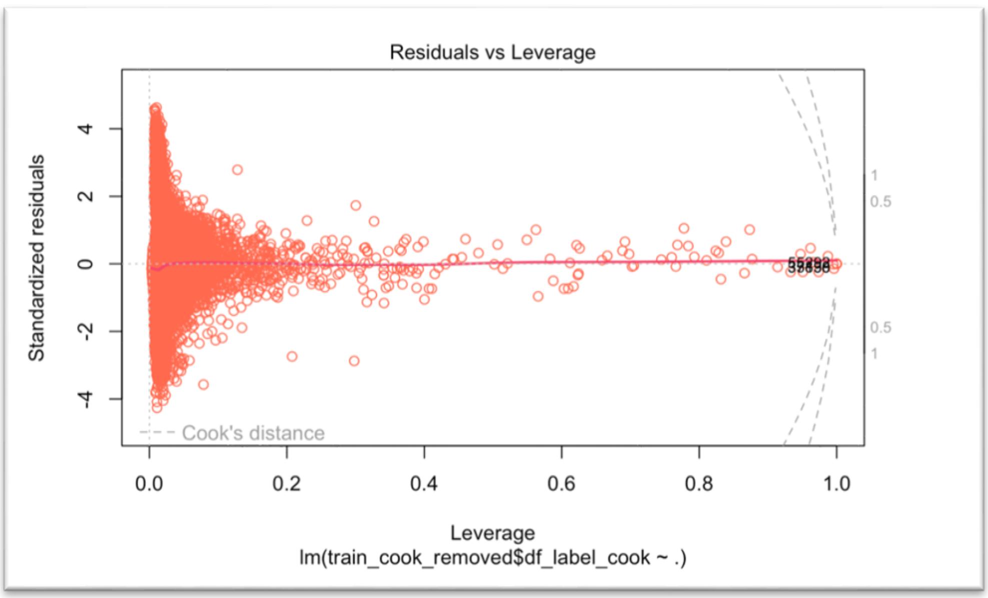
```
"The classification error rate for the training set is:
0.0909514459715798"
```

```
plot(fit_outliers, lwd = '2', col = 'coral1')
```









Let's compare the classification error rates with the outliers removed to when the outliers were included.

Testing Set Error rates

With outliers: 0.09408

Without outliers: 0.09154

Training Set Error rates

With outliers: 0.09364

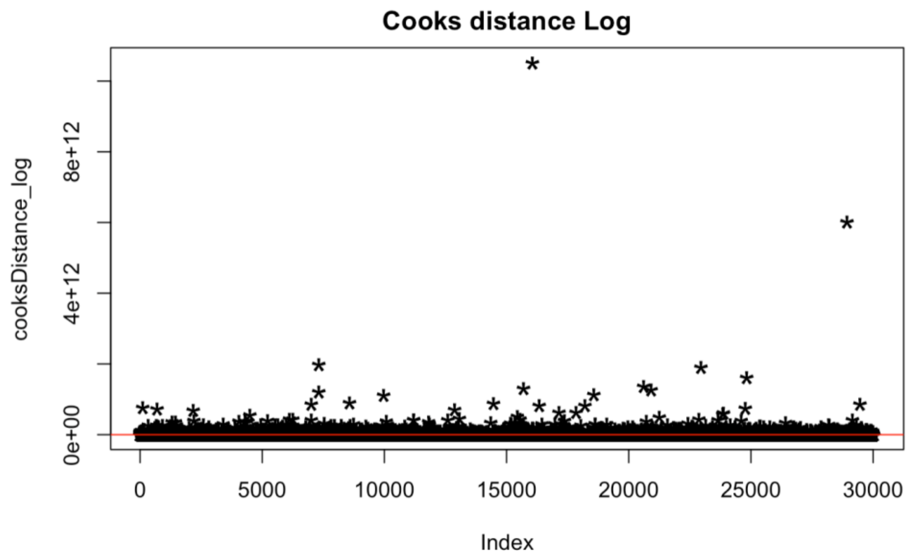
Without outliers: 0.09095

After performing cooks distance and removing the outliers, the errors rates for both the training and testing decreased by a small margin. Due to the outliers being removed, it is expected that the error rates would decrease.

Removing outliers with cooks distance (Logistic Regression on K vs not K)

The above process is repeated but with Logistic Regression. The same train_cook data set is used to fit the model and remove outliers.

```
cook_log <- glm(as.factor(df_label_cook) ~ ., data = train_cook, family = 'binomial')

cooksDistance_log <- cooks.distance(cook_log)

sample_size <- nrow(train_cook)
plot(cooksDistance_log, pch="*", cex=2, main="Cooks distance Log")
abline(h = 4/sample_size, col="red")

n <- nrow(train_cook)
outliers <- as.numeric(names(cooksDistance_log)[(cooksDistance_log > (4/n))])
index <- data.frame(outliers)
train_cook_removed_log <- train_cook[!(row.names(train_cook) %in% index$outliers),]
cook_log_removed <- glm(as.factor(df_label_cook) ~ ., data = train_cook_removed_log,
family = 'binomial')
```




```
predicted_cook <- predict(cook_log_removed, test_cook, type="response")
p_class_cook <- ifelse(predicted_cook >.5,"1","-1")
confusion_mat_test_cook <- table(p_class_cook, test_cook[['df_label_cook']])
confusion_mat_test_cook
```

```
p_class_cook     -1      1
          -1  26349    740
          1     640   2222
```

```
predicted_cook <- predict(cook_log_removed, train_cook, type="response")
p_class_cook <- ifelse(predicted_cook >.5,"1","-1")
confusion_mat_train_cook <- table(p_class_cook, train_cook[['df_label_cook']])
confusion_mat_train_cook
```

```
p_class_cook     -1      1
          -1  26612    631
          1     450   2356
```

```
TP <- confusion_mat_test_cook[1,1]
TN <- confusion_mat_test_cook[2,2]
FP <- confusion_mat_test_cook[1,2]
FN <- confusion_mat_test_cook[2,1]
Accuracy_test <- (TP + TN) / (TP + FP + TN + FN)
classification_error_test <- 1-Accuracy_test
paste0('The classification error rate for the testing set is:
',classification_error_test)
```

```
"The classification error rate for the testing set is:
0.046075256251878"
```

```
TP <- confusion_mat_train_cook[1,1]
TN <- confusion_mat_train_cook[2,2]
FP <- confusion_mat_train_cook[1,2]
FN <- confusion_mat_train_cook[2,1]
Accuracy <- (TP + TN) / (TP + FP + TN + FN)
classification_error_train <- 1-Accuracy
paste0('The classification error rate for the training set is:
',classification_error_train)
```

```
"The classification error rate for the training set is:
0.0359745748610603"
```

Let's compare the classification error rates with the outliers removed to when the outliers were included.

Testing Set Error rates

With outliers: 0.04607

Without outliers: 0.04607

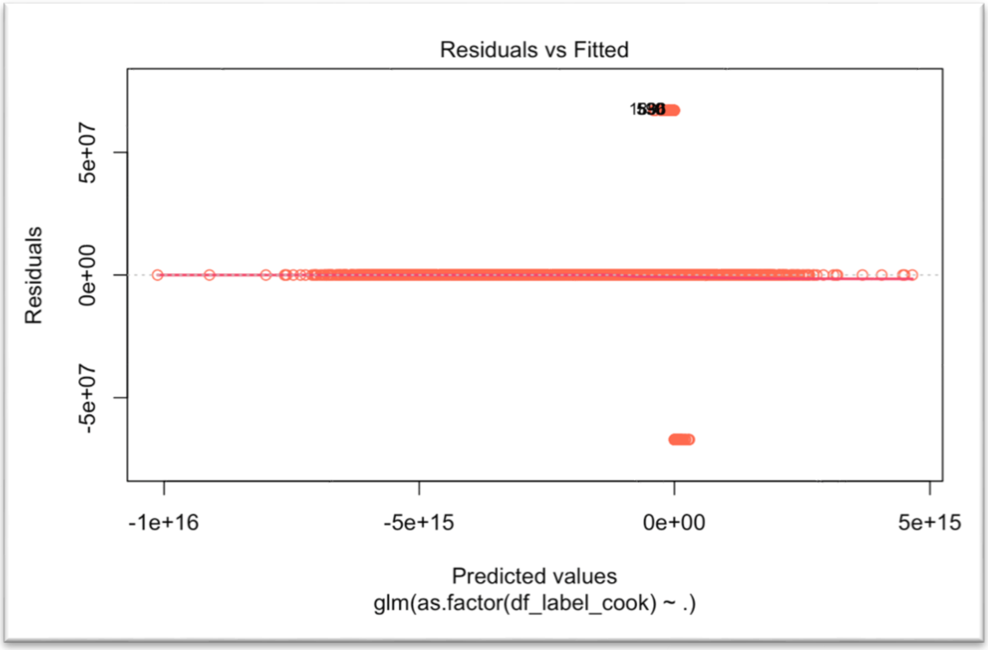
Training Set Error rates

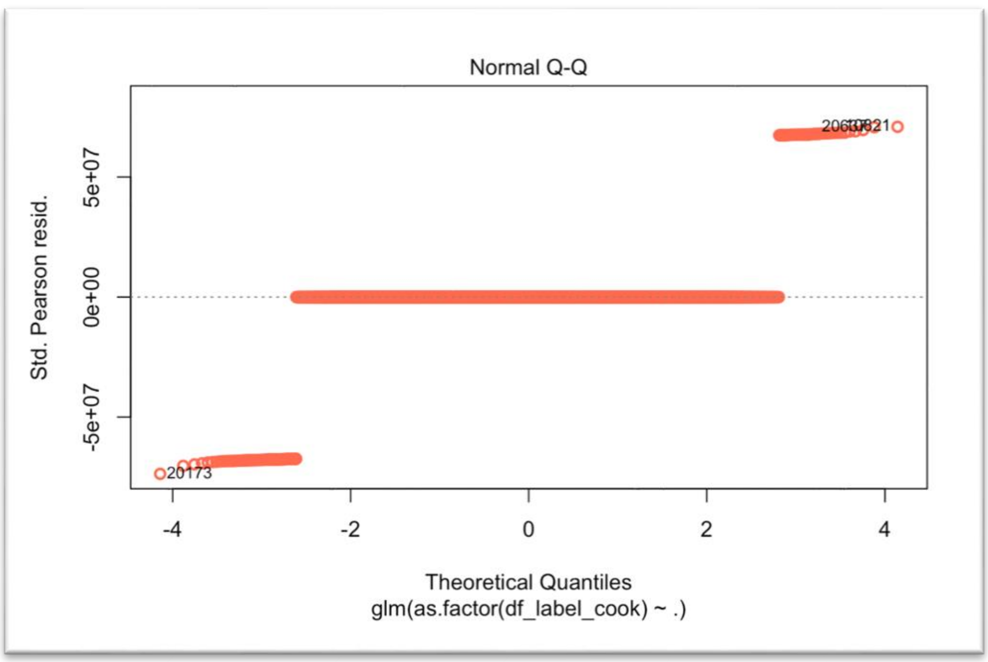
With outliers: 0.0384

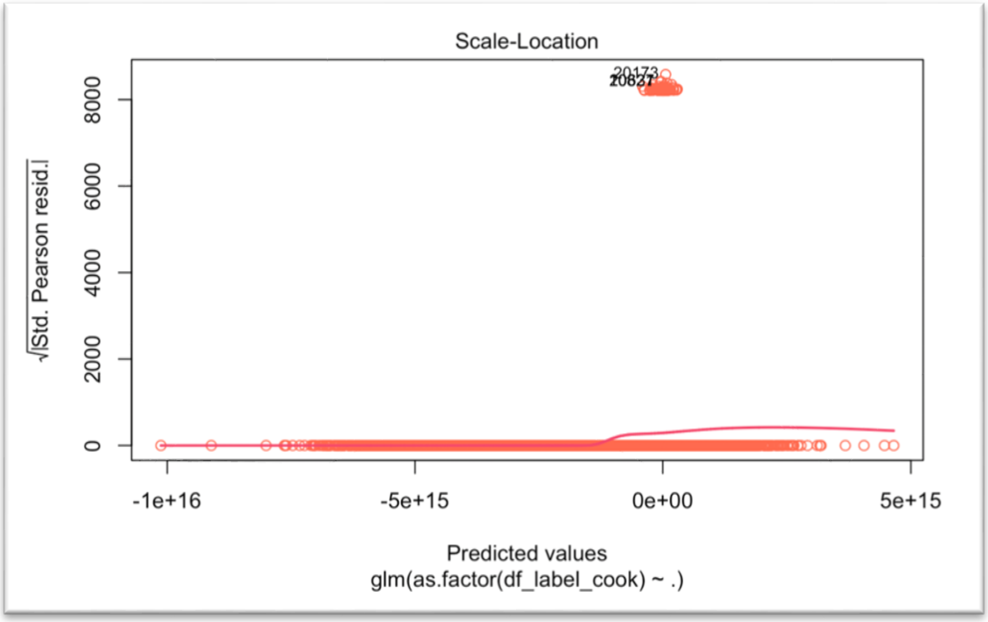
Without outliers: 0.03597

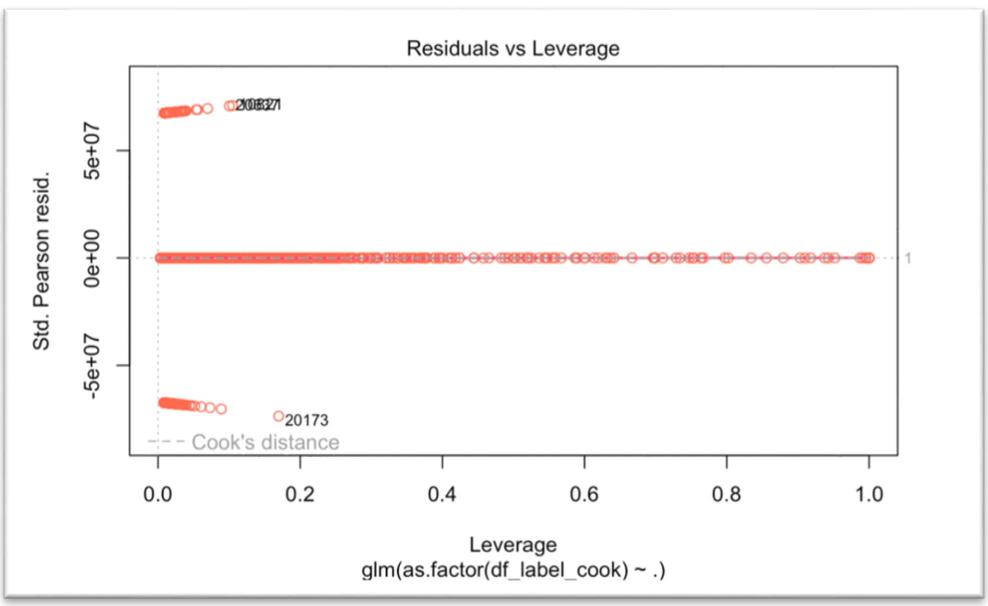
After performing cooks distance and removing the outliers, the errors rates for the training set decreased, which was expected because the outliers being removed would have made the fit more accurate. However, the testing error rate stayed exactly the same.

```
plot(cook_log_removed, lwd = '2', col = 'coral1')
```


Residuals vs Fitted
1590
5e+07
0e+00
-5e+07
Residuals
-1e+16
-5e+15
0e+00
5e+15
Predicted values
glm(as.factor(df_label_cook) ~ .)


Normal Q-Q
5e+07
0e+00
-5e+07
Std. Pearson resid.
20173
-4
-2
0
2
4
Theoretical Quantiles
glm(as.factor(df_label_cook) ~ .)





Repeating part 2

Part 2 will be repeated with all digit pairs with the outliers removed. This is for linear regression.

```
label = list()
 list_df = list()
 list_of_pairs = list()
 for (x in 0:9){
   corr = x
   list_to_nine <- list(0,1,2,3,4,5,6,7,8,9)
   second_loop_list <- list_to_nine[-c(corr+1)]

  for (y in 0:9) {
     if (y > x) {
       incorr = y
       pair = list()
       pair <- append(pair,corr)
       pair <- append(pair,incorr)
       list_of_pairs[[length(list_of_pairs) + 1]] <- pair
       df_label2 <- replace(train_Labels, which((train_Labels == corr) %in% TRUE), 1)
       df_label2 <- replace(df_label2, which((train_Labels == incorr) %in% TRUE), -1)
       list_df[[length(list_df) + 1]] <- df_label2
     }
   }
 }
list_of_df_lin = list()
 for (one_pair in list_df){
   df_y2 <- data.frame(one_pair)
   df2 <- cbind(train_digits, df_y2)
   df2 <- subset(df2,(one_pair==1|one_pair==-1))
   list_of_df_lin[[length(list_of_df_lin) + 1]] <- df2

}
```

The code has been modified to add cooks distance and remove the outliers.

```
set.seed(1)
 train_error <- list()
 test_error <- list()
 for (df_for_split in list_of_df_lin){

   set.seed(1)

  sample <- sample(c(TRUE, FALSE), nrow(df_for_split), replace=TRUE, prob=c(0.5,0.5))
   train_cook  <- df_for_split[sample, ]
   test_cook   <- df_for_split[!sample, ]
   fit_train01 <- lm(train_cook$one_pair ~ ., data = train_cook)
   cooksDistance <- cooks.distance(fit_train01)
   n <- nrow(train_cook)
   outliers <- as.numeric(names(cooksDistance)[(cooksDistance > (4/n))])
   index <- data.frame(outliers)
   train_cook_removed <- train_cook[!(row.names(train_cook) %in% index$outliers),]
   fit_outliers <- lm(train_cook_removed$one_pair ~ ., data = train_cook_removed)

  predicted01 <- predict(fit_outliers, test_cook, type="response")
   p_class01 <- ifelse(predicted01 >.5,"1","-1")
   confusion_mat_test <- table(p_class01, test_cook[['one_pair']])



  predicted <- predict(fit_outliers, train_cook, type="response")
   p_class <- ifelse(predicted >.5,"1","-1")
   confusion_mat_train <- table(p_class, train_cook[['one_pair']])



  TP <- confusion_mat_test[1,1]
   TN <- confusion_mat_test[2,2]
   FP <- confusion_mat_test[1,2]
   FN <- confusion_mat_test[2,1]
   Accuracy_test <- (TP + TN) / (TP + FP + TN + FN)
```

```
    classification_error_test <- 1-Accuracy_test
    test_error[[length(test_error) + 1]] <- classification_error_test



   TP <- confusion_mat_train[1,1]
    TN <- confusion_mat_train[2,2]
    FP <- confusion_mat_train[1,2]
    FN <- confusion_mat_train[2,1]
    Accuracy <- (TP + TN) / (TP + FP + TN + FN)
    classification_error_train <- 1-Accuracy
    train_error[[length(train_error) + 1]] <- classification_error_train



}
```

```
suppressWarnings({

my_mat_log <- matrix(, ncol = 10, nrow=10)


my_mat_log[lower.tri(my_mat_log, diag = FALSE)] <- test_error[1:45]
my_mat_log <- matrix(my_mat_log, ncol = 10, nrow=10)
 my_mat_log[upper.tri(my_mat_log, diag = FALSE)] <- train_error[1:45]
my_mat_log_1 <- rbind(c(0:9), my_mat_log)
 my_mat_log_2 <- cbind(c(-1:9), my_mat_log_1)
 my_mat_log_2

})
```

-1	0	1	2	3	4	5	6	7	8	9
0	NA	0.0154434	0.04168016	0.03228261	0.02069185	0.02336156	0.03847387	0.03547977	0.07940766	0.04519651
1	0.0214599	NA	0.03987173	0.06054535	0.06431287	0.01572529	0.0217633	0.06689072	0.05357524	0.0892318
2	0.05137791	0.0275086	NA	0.04792816	0.02733067	0.0236755	0.06548635	0.03664583	0.03546258	0.05013339
3	0.04914734	0.0247883	0.07868852	NA	0.02364329	0.01576684	0.04338395	0.03564144	0.02271986	0.01666313
4	0.03825078	0.01967884	0.04795108	0.04319149	NA	0.02201209	0.05300589	0.07790493	0.04208395	0.03965667
5	0.07076145	0.02956067	0.05919754	0.0849446	0.05018751	NA	0.04990279	0.03453807	0.04584152	0.01728982
6	0.05246253	0.0205297	0.0525641	0.04693294	0.02880791	0.05132432	NA	0.05238845	0.07130454	0.04111632
7	0.02703556	0.02471019	0.04284364	0.06106314	0.0510409	0.05405697	0.02062507	NA	0.04041416	0.06443408
8	0.0743207	0.0465775	0.08234159	0.09229133	0.04895934	0.1100209	0.04641668	0.04234424	NA	0.05661605
9	0.03131346	0.02470294	0.04119091	0.05861193	0.07232536	0.05901711	0.02490647	0.07773109	0.06500751	NA

I expected the error rates for both the training and testing datasets to decrease. Compared to the error rate matrix with outliers, most of the error rates did decrease. For example, the highest error rate was 0.12 the pair 5 and 8. With the outliers removed, the error rate is 0.11. The pair 5 and 8 still has the highest error rate in the testing data set.

The training error rates also mostly decrease with a few exceptions. The lowest error rate in the training set is 0.015 for the pair 0 and 1. This is the pair that had the lowest error rate before when the outliers were included.

```
set.seed(1)
train_error <- list()
test_error <- list()
for (df_for_split in list_of_df_lin){
```

```
  set.seed(1)

  sample <- sample(c(TRUE, FALSE), nrow(df_for_split), replace=TRUE, prob=c(0.5,0.5))
  train_cook  <- df_for_split[sample, ]
  test_cook   <- df_for_split[!sample, ]
  fit_train01 <- glm(as.factor(one_pair) ~ ., data = train_cook, family = 'binomial')
  cooksDistance <- cooks.distance(fit_train01)
  outliers <- as.numeric(names(cooksDistance)[(cooksDistance > (4/n))])
  index <- data.frame(outliers)
  train_cook_removed <- train_cook[!(row.names(train_cook) %in% index$outliers),]

  fit_outliers <- glm(as.factor(one_pair) ~ ., data = train_cook_removed, family = 'binomial')


  predicted01 <- predict(fit_outliers, test_cook, type="response")
  p_class01 <- ifelse(predicted01 >.5,"1","-1")
  confusion_mat_test <- table(p_class01, test_cook[['one_pair']])


  predicted <- predict(fit_outliers, train_cook, type="response")
  p_class <- ifelse(predicted >.5,"1","-1")
  confusion_mat_train <- table(p_class, train_cook[['one_pair']])


  TP <- confusion_mat_test[1,1]
  TN <- confusion_mat_test[2,2]
  FP <- confusion_mat_test[1,2]
  FN <- confusion_mat_test[2,1]
  Accuracy_test <- (TP + TN) / (TP + FP + TN + FN)
  classification_error_test <- 1-Accuracy_test
  test_error[[length(test_error) + 1]] <- classification_error_test


  TP <- confusion_mat_train[1,1]
  TN <- confusion_mat_train[2,2]
  FP <- confusion_mat_train[1,2]
  FN <- confusion_mat_train[2,1]
  Accuracy <- (TP + TN) / (TP + FP + TN + FN)
  classification_error_train <- 1-Accuracy
  train_error[[length(train_error) + 1]] <- classification_error_train


}
```

```
suppressWarnings({

my_mat_log <- matrix(, ncol = 10, nrow=10)

my_mat_log[lower.tri(my_mat_log, diag = FALSE)] <- test_error[1:45]
my_mat_log <- matrix(my_mat_log, ncol = 10, nrow=10)
my_mat_log[upper.tri(my_mat_log, diag = FALSE)] <- train_error[1:45]
my_mat_log_1 <- rbind(c(0:9), my_mat_log)
my_mat_log_2 <- cbind(c(-1:9), my_mat_log_1)
my_mat_log_2

})
```

-1	0	1	2	3	4	5	6	7	8	9
0	NA	0.001273683	0.06090055	0.001195652	0.001379457	0.003606146	0.09073421	0.005189578	0.04614229	0.001965066
1	0.006735589	NA	0.006520577	0.009571508	0.02053232	0.001444159	0.00127085	0.03024718	0.002774813	0.05233143
2	0.08449049	0.03454204	NA	0.0009737098	0.001404343	0.001490066	0.01397184	0.001940073	0.001006824	0.002890173
3	0.03294143	0.02232487	0.04791116	NA	0.003173596	0.00159261	0.003036876	0.0009661836	0.002391564	0.001061346
4	0.01998496	0.01904912	0.03003647	0.01925532	NA	0.003100295	0.006000667	0.1077245	0.003942041	0.008257279
5	0.04373146	0.02025151	0.0340934	0.1247013	0.0346349	NA	0.002808382	0.001283148	0.01036892	0.0006483683
6	0.02944325	0.01676853	0.03098291	0.01875199	0.03364506	0.03912518	NA	0.002309711	0.01258954	0.0118236
7	0.02503682	0.02516779	0.0305576	0.03900967	0.04163584	0.02826931	0.02104599	NA	0.002672011	0.01632047
8	0.05359252	0.1114083	0.06786748	0.07995747	0.04734174	0.08146842	0.04491243	0.04139388	NA	0.005097614
9	0.02629048	0.02095059	0.02966599	0.03872196	0.05172229	0.04113646	0.0242651	0.05588235	0.02821283	NA

For the logistic error rate matrix, most of the error rates decreased after removing the outliers. However, there are a few that increased. The lowest error rate for testing is the pair 0 and 1. The highest error rate for the testing was for pairs 3 and 5. Before, the training set had a lot of 0s as error rates, which was unexpected. Here none of the error rates are 0, however they are still very small.