CS109b-hw4-submission

Ihsaan Patel

March 4, 2017

Libraries

```
library(e1071)
library(caret)
library(mclust)
library(MCMCpack)
```

Problem 1: Celestial Object Classification

Load Data

```
# Load training set
dataset_1_train <- read.table("datasets/dataset_1_train.txt", header = TRUE, sep = ",")
dataset_1_train$Class <- as.factor(dataset_1_train$Class)

# Load testing set
dataset_1_test <- read.table("datasets/dataset_1_test.txt", header = TRUE, sep = ",")
dataset_1_test$Class <- as.factor(dataset_1_test$Class)
```

1. RBF Kernel: Gamma = 1, Cost = 1

```
# Fit SVM
model.svm_rbf_g1_c1 <- svm(Class ~ ., data = dataset_1_train, cost = 1, gamma = 1, kernel = "radial")

# Predict Test Set and Calculate Misclassification Rate
misclassification_rate.svm_rbf_g1_c1 <- classError(predict(model.svm_rbf_g1_c1, dataset_1_test), dataset
print(sprintf("SVM Class = 1 Gamma = 1 Misclassification Rate: %.4f", misclassification_rate.svm_rbf_g1_
```

```
## [1] "SVM Class = 1 Gamma = 1 Misclassification Rate: 0.2768"
```

2. Confusion Matrix

Train Confusion Matrix

```
# Print Train Confusion Matrix
confusionMatrix(table(predict(model.svm_rbf_g1_c1, dataset_1_train), dataset_1_train$Class))
```

```
## Confusion Matrix and Statistics
##
##
##       1   2   3   4
```

```
##   1  44   0   0   0
##   2   0  72   0   0
##   3   0   0 492   0
##   4   0   0   0  81
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9947, 1)
##     No Information Rate : 0.7141
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity           1.00000   1.0000   1.0000   1.0000
## Specificity           1.00000   1.0000   1.0000   1.0000
## Pos Pred Value        1.00000   1.0000   1.0000   1.0000
## Neg Pred Value        1.00000   1.0000   1.0000   1.0000
## Prevalence            0.06386   0.1045   0.7141   0.1176
## Detection Rate        0.06386   0.1045   0.7141   0.1176
## Detection Prevalence  0.06386   0.1045   0.7141   0.1176
## Balanced Accuracy     1.00000   1.0000   1.0000   1.0000
```

Test Confusion Matrix

```
# Print Test Confusion Matrix
confusionMatrix(table(predict(model.svm_rbf_g1_c1, dataset_1_test), dataset_1_test$Class))
```

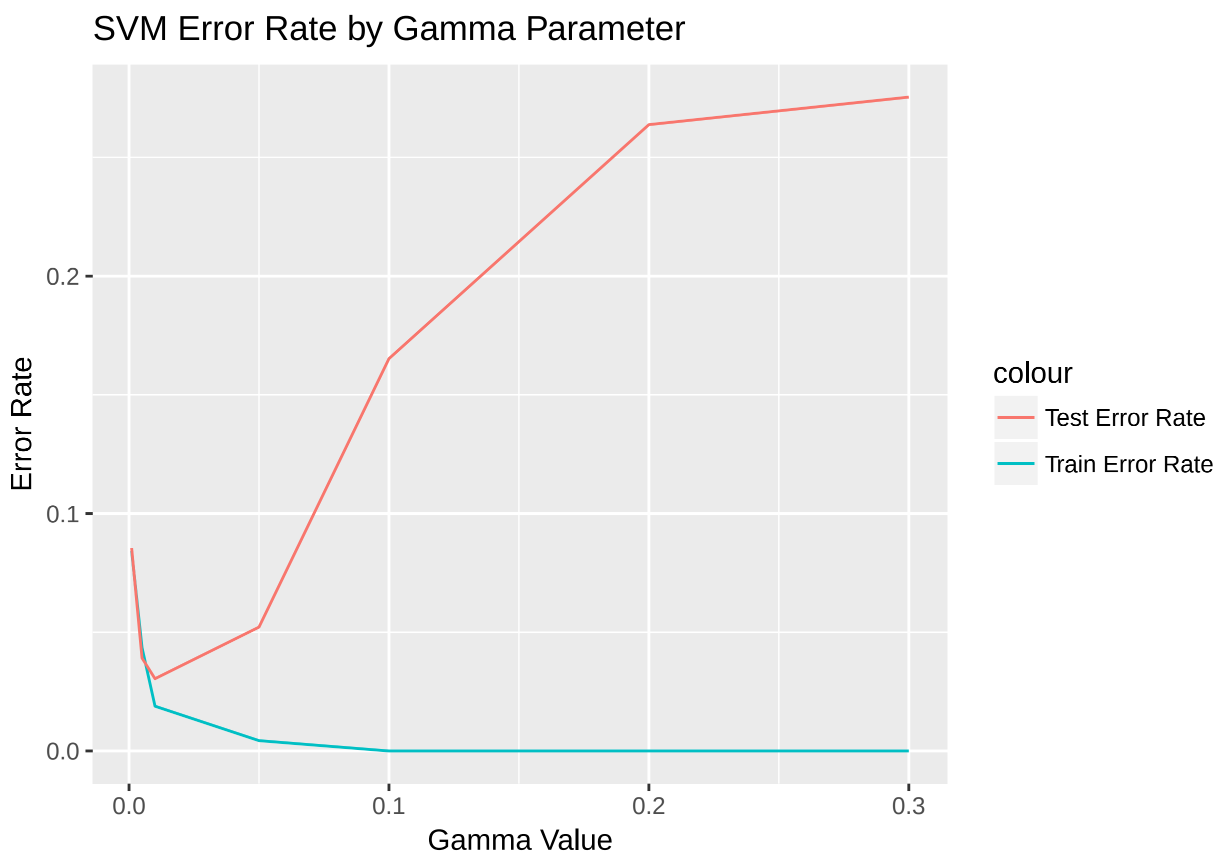
```
## Confusion Matrix and Statistics
##
##
##       1   2   3   4
##   1   0   0   0   0
##   2   0   0   0   0
##   3  46  73 499  72
##   4   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.7232
##                  95% CI : (0.6882, 0.7563)
##     No Information Rate : 0.7232
##     P-Value [Acc > NIR] : 0.5195
##
##                   Kappa : 0
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity           0.00000   0.0000   1.0000   0.0000
## Specificity           1.00000   1.0000   0.0000   1.0000
## Pos Pred Value            NaN      NaN   0.7232      NaN
## Neg Pred Value        0.93333   0.8942      NaN   0.8957
## Prevalence            0.06667   0.1058   0.7232   0.1043
## Detection Rate        0.00000   0.0000   0.7232   0.0000
## Detection Prevalence  0.00000   0.0000   1.0000   0.0000
## Balanced Accuracy     0.50000   0.5000   0.5000   0.5000
```

The model appears to over-fit the training set by correctly predicting the class for every observation, while for the testing set the model just predicts class 3 for every observation.

3. Tuning Gamma

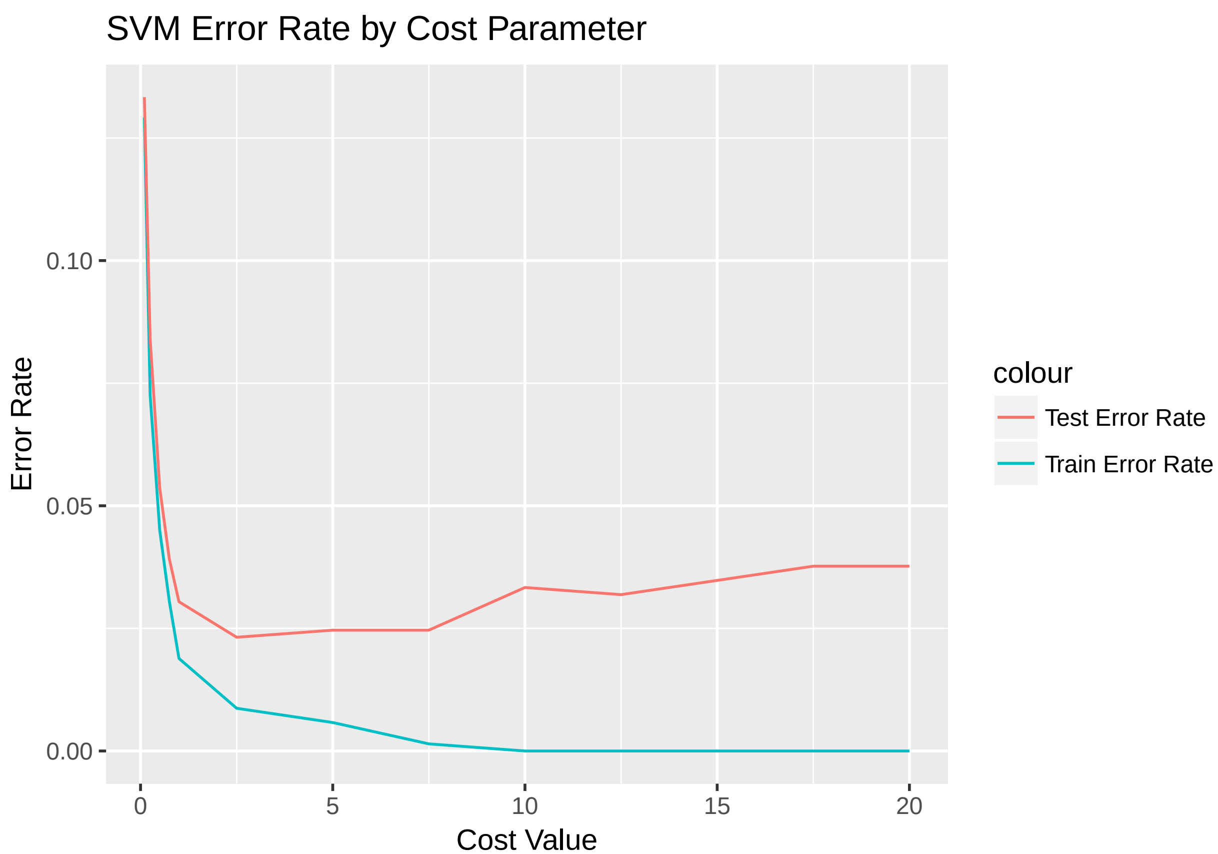
```r
# Initiate list with gammas and lists to store error rates
gammas <- c(0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.3)
gamma_train_scores <- rep(0., length(gammas))
gamma_test_scores <- rep(0., length(gammas))

# Loop through each gamma value, fitting an svm and calculating training and test error artes
for (gamma in 1:length(gammas)) {
  model.svm_tune_gamma <- svm(Class ~ ., data = dataset_1_train, cost = 1, gamma = gammas[gamma], kernel
  gamma_train_scores[gamma] <- classError(predict(model.svm_tune_gamma, dataset_1_train), dataset_1_trai
  gamma_test_scores[gamma] <- classError(predict(model.svm_tune_gamma, dataset_1_test), dataset_1_test$C
}

# Plot training and test error rates
gamma_scores <- data.frame(gammas = gammas, train_scores = gamma_train_scores, test_scores = gamma_test_
ggplot(gamma_scores, aes(gammas)) + geom_line(aes(y = train_scores, colour = "Train Error Rate")) + geom
```



It looks like model performance on the test set improves at very low levels as gamma increases (until gamma = 0.01) but then deteriorates rapidly as gamma continues to increase. The train error rate continues to improve as gamma increases, indicating clear over-fitting post gamma = 0.01.

4. Tuning Class

```
# Initiate list with costs and lists to store error rates
costs <- c(0.1, 0.25, 0.5, 0.75, 1., 2.5, 5., 7.5, 10., 12.5, 15., 17.5, 20.)
cost_train_scores <- rep(0., length(costs))
cost_test_scores <- rep(0., length(costs))

# Loop through each cost value, fitting an svm and calculating training and test error artes
for (cost in 1:length(costs)) {
  model.svm_tune_cost <- svm(Class ~ ., data = dataset_1_train, cost = costs[cost], gamma = 0.01, kernel
  cost_train_scores[cost] <- classError(predict(model.svm_tune_cost, dataset_1_train), dataset_1_train$C
  cost_test_scores[cost] <- classError(predict(model.svm_tune_cost, dataset_1_test), dataset_1_test$Clas
}

# Plot training and test error rates
cost_scores <- data.frame(costs = costs, train_scores = cost_train_scores, test_scores = cost_test_score
ggplot(cost_scores, aes(costs)) + geom_line(aes(y = train_scores, colour = "Train Error Rate")) + geom_l
```



It looks like model performance on the test set improves at low levels as cost increases (until cost = 2.5) but then deteriorates slowly as cost continues to increase. The train error rate continues to improve as cost increases, indicating clear over-fitting post cost = 2.5.

5. Tune Various SVM Models

```
costs <- 10^c(-10:10)
gammas <- 10^c(-10:10)
```

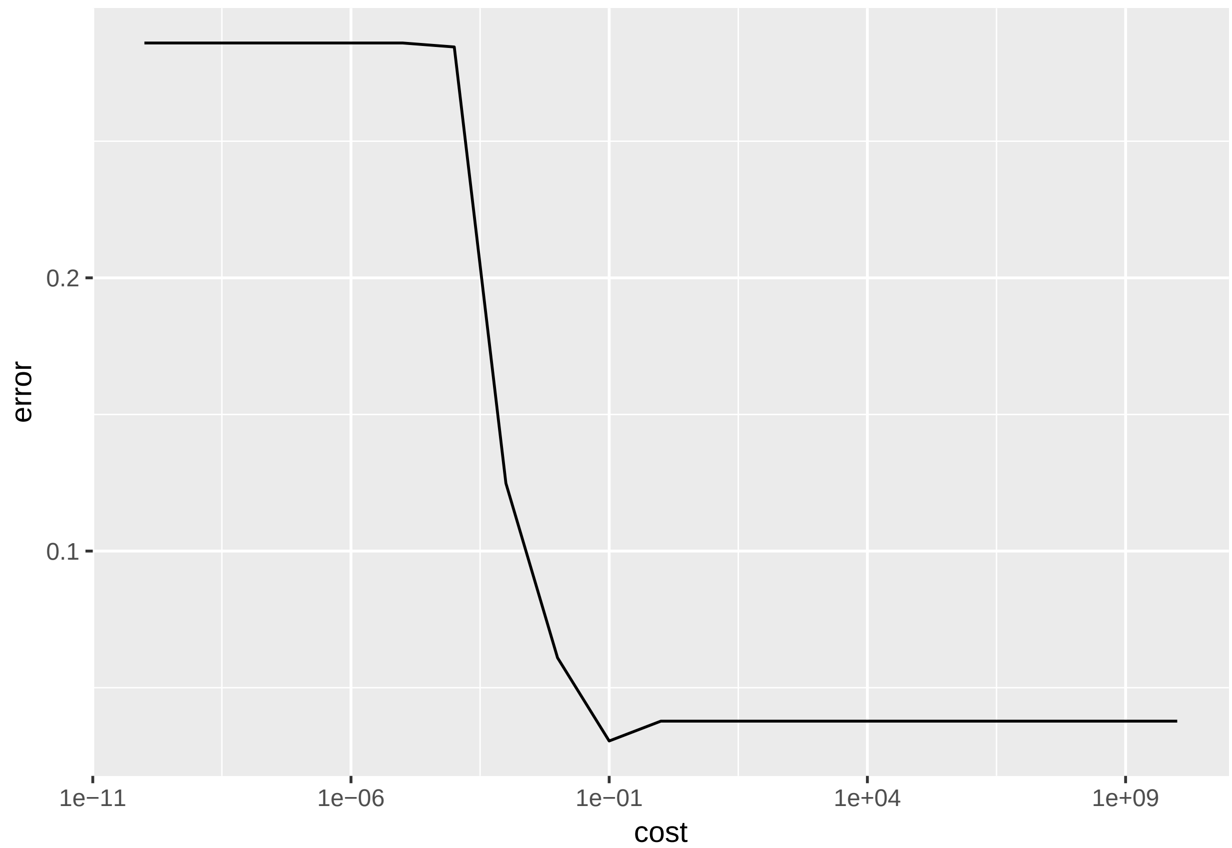
Linear Model

```
# Tune SVM Linear Model
svm.linear.tune <- tune(svm, Class ~ ., data = dataset_1_train, ranges = list(cost = costs, kernel = 'li

# Print tune output
svm.linear.tune
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 5-fold cross validation
##
## - best parameters:
##  cost kernel
```

```
##   0.1 linear
##
## - best performance: 0.03048768
```

```r
# Plot error rate over tuning parameters
ggplot(svm.linear.tune$performances, mapping = aes(x = cost, y = error)) + geom_line() + scale_x_log10(
```




```r
svm.linear.tune_error_rate <- classError(predict(svm.linear.tune$best.model, dataset_1_test), dataset_1_

# Print misclassification rate
sprintf("SVM Linear Misclassification Rate: %.4f", svm.linear.tune_error_rate)
```

```
## [1] "SVM Linear Misclassification Rate: 0.0145"
```

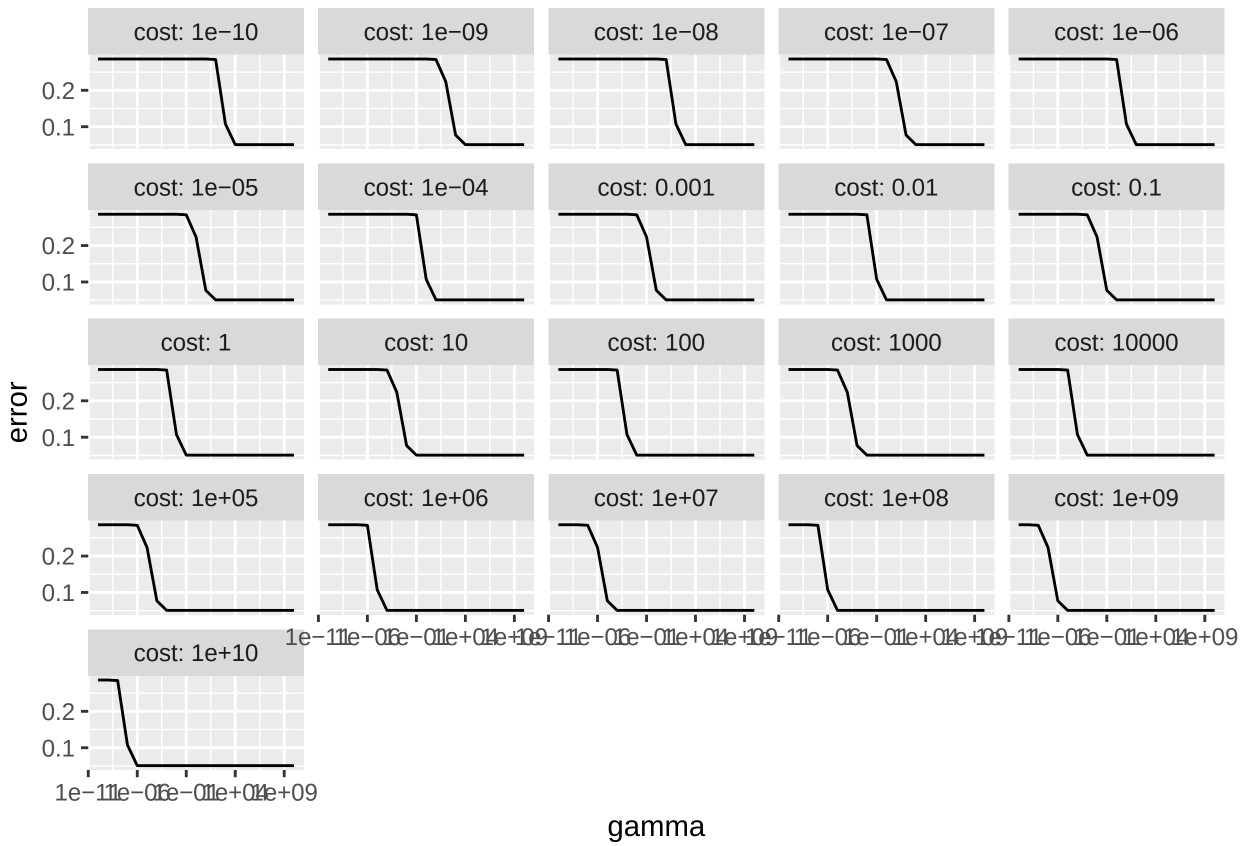
Poly Model

```r
# Tune SVM Poly Model
svm.poly.tune <- tune(svm, Class ~ ., data = dataset_1_train, ranges = list(gamma = gammas, cost = costs

# Print tune output
svm.poly.tune
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 5-fold cross validation
```

```
##
## - best parameters:
##  gamma  cost     kernel degree
##  10000 1e-10 polynomial      2
##
## - best performance: 0.05078811
```

```r
# Plot error rate over tuning parameters
ggplot(svm.poly.tune$performances, mapping = aes(x = gamma, y = error)) + geom_line() + facet_wrap(~cos
```




```r
# Print misclassification rate
svm.poly.tune_error_rate <- classError(predict(svm.poly.tune$best.model, dataset_1_test), dataset_1_tes
sprintf("SVM Poly Misclassification Rate: %.4f", svm.poly.tune_error_rate)
```

```
## [1] "SVM Poly Misclassification Rate: 0.0580"
```

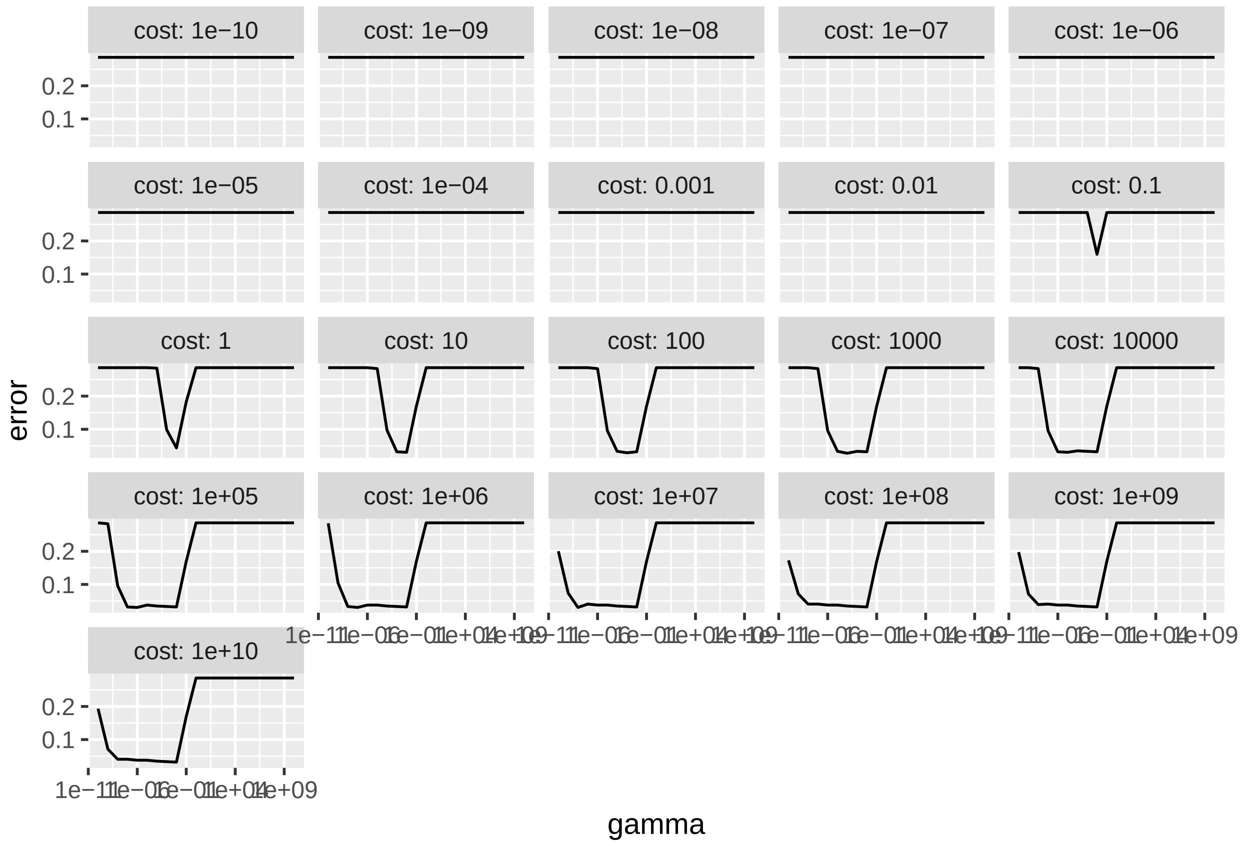
RBF Model

```r
# Tune SVM RBF Model
svm.rbf.tune <- tune(svm, Class ~ ., data = dataset_1_train, ranges = list(gamma = gammas, cost = costs

# Print tune output
svm.rbf.tune
```

```
##
## Parameter tuning of 'svm':
##
```

```
## - sampling method: 5-fold cross validation
##
## - best parameters:
##  gamma cost kernel
##  1e-04 1000 radial
##
## - best performance: 0.02757855
```

```
# Plot error rate over tuning parameters
ggplot(svm.rbf.tune$performances, mapping = aes(x = gamma, y = error)) + geom_line() + facet_wrap(~cost
```




```
# Print misclassification rate
svm.rbf.tune_error_rate <- classError(predict(svm.rbf.tune$best.model, dataset_1_test), dataset_1_test$
sprintf("SVM RBF Misclassification Rate: %.4f", svm.rbf.tune_error_rate)
```

```
## [1] "SVM RBF Misclassification Rate: 0.0188"
```

6. Best Model

The linear is the best in terms of model accuracy as it has a lower misclassification rate on the test set when compared to the tuned polynomial and radial-basis models. It also performs much better than a naive classifier that predicts the most common class on all points, which is what the original SVM RBF Model with cost = 1 and gamma = 1 also did.

Problem 2: Return of the Bayesian Hierarchical Modedl

```r
# Load dataset
dataset_2 <- read.table("datasets/dataset_2.txt", header = TRUE, sep = ",")
```

1(a) Pooled Model

```r
model.pooled <- glm(contraceptive_use ~ living.children, data = dataset_2, family = binomial(link = "lo
summary(model.pooled)
```

```
##
## Call:
## glm(formula = contraceptive_use ~ living.children, family = binomial(link = "logit"),
##     data = dataset_2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1109  -1.0245  -0.8631   1.2454   1.5285
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -1.00804    0.11413  -8.832  < 2e-16 ***
## living.children  0.21240    0.03816   5.565 2.61e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2590.9  on 1933  degrees of freedom
## Residual deviance: 2559.4  on 1932  degrees of freedom
## AIC: 2563.4
##
## Number of Fisher Scoring iterations: 4
```

The number of living children increases the probability that a women uses contraception and this association is statistically significant at the <.001 level.

1(b) Unpooled Model

```r
model.unpooled <- glm(contraceptive_use ~ -1 + living.children*as.factor(district), data = dataset_2, f
summary(model.unpooled)
```

```
##
## Call:
## glm(formula = contraceptive_use ~ -1 + living.children * as.factor(district),
##     family = binomial(link = "logit"), data = dataset_2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7816  -0.9576  -0.6271   1.1045   2.2425
##
```

```
## Coefficients:
##                                          Estimate Std. Error z value
## living.children                         1.064e-01  1.806e-01   0.589
## as.factor(district)101                 -1.376e+00  5.758e-01  -2.390
## as.factor(district)102                 -2.062e+00  1.543e+00  -1.336
## as.factor(district)103                  1.657e+01  3.298e+03   0.005
## as.factor(district)104                 -8.343e-01  1.090e+00  -0.765
## as.factor(district)105                 -8.590e-01  8.083e-01  -1.063
## as.factor(district)106                 -1.343e+00  6.998e-01  -1.919
## as.factor(district)107                 -6.117e-01  1.164e+00  -0.525
## as.factor(district)108                 -1.924e+00  9.073e-01  -2.121
## as.factor(district)109                 -1.679e+00  1.544e+00  -1.088
## as.factor(district)110                 -1.891e+00  2.049e+00  -0.923
## as.factor(district)111                 -1.657e+01  1.042e+03  -0.016
## as.factor(district)112                  4.259e-01  1.106e+00   0.385
## as.factor(district)113                 -1.920e+00  1.061e+00  -1.810
## as.factor(district)114                 -7.132e-02  4.190e-01  -0.170
## as.factor(district)115                 -2.022e+00  1.130e+00  -1.790
## as.factor(district)116                 -7.041e-01  9.589e-01  -0.734
## as.factor(district)117                 -5.155e+00  2.996e+00  -1.721
## as.factor(district)118                 -9.284e-01  6.899e-01  -1.346
## as.factor(district)119                 -1.662e+00  9.771e-01  -1.701
## as.factor(district)120                 -4.055e-01  1.083e+00  -0.374
## as.factor(district)121                  2.092e+00  1.275e+00   1.642
## as.factor(district)122                 -2.621e-01  1.442e+00  -0.182
## as.factor(district)123                 -3.545e+00  2.550e+00  -1.390
## as.factor(district)124                 -2.791e+01  1.432e+03  -0.019
## as.factor(district)125                 -2.661e+00  7.653e-01  -3.477
## as.factor(district)126                 -1.285e+00  1.881e+00  -0.683
## as.factor(district)127                 -1.524e+00  8.618e-01  -1.769
## as.factor(district)128                 -2.792e+00  1.122e+00  -2.489
## as.factor(district)129                 -2.409e+00  9.649e-01  -2.496
## as.factor(district)130                 -2.150e+00  6.768e-01  -3.177
## as.factor(district)131                 -7.139e-01  8.372e-01  -0.853
## as.factor(district)132                 -1.982e+00  1.676e+00  -1.183
## as.factor(district)133                 -5.714e+00  3.168e+00  -1.804
## as.factor(district)134                  4.512e-01  1.006e+00   0.449
## as.factor(district)135                 -5.451e-01  7.747e-01  -0.704
## as.factor(district)136                 -6.456e-01  1.468e+00  -0.440
## as.factor(district)137                  9.290e-01  1.665e+00   0.558
## as.factor(district)138                 -2.896e+01  1.866e+03  -0.016
## as.factor(district)139                  7.588e-01  8.597e-01   0.883
## as.factor(district)140                 -7.386e-02  6.892e-01  -0.107
## as.factor(district)141                 -1.754e-01  1.068e+00  -0.164
## as.factor(district)142                 -3.082e+01  1.554e+03  -0.020
## as.factor(district)143                 -8.442e-01  6.719e-01  -1.256
## as.factor(district)144                 -2.262e+00  1.263e+00  -1.791
## as.factor(district)145                 -1.942e+00  1.003e+00  -1.937
## as.factor(district)146                 -3.495e-01  5.370e-01  -0.651
## as.factor(district)147                 -2.249e+00  1.468e+00  -1.532
## as.factor(district)148                 -5.282e-01  7.161e-01  -0.738
## as.factor(district)149                 -1.657e+01  2.013e+03  -0.008
## as.factor(district)150                 -2.081e+00  1.182e+00  -1.761
## as.factor(district)151                  1.211e-01  8.387e-01   0.144
```

```
## as.factor(district)152                    -1.134e+00  6.188e-01  -1.833
## as.factor(district)153                    -2.305e+00  1.305e+00  -1.766
## as.factor(district)154                     4.225e+01  1.446e+03   0.029
## as.factor(district)155                     2.891e-01  6.473e-01   0.447
## as.factor(district)156                    -3.106e+00  1.590e+00  -1.954
## as.factor(district)157                    -6.170e-01  7.906e-01  -0.780
## as.factor(district)158                    -2.687e+00  2.997e+00  -0.897
## as.factor(district)159                    -2.744e+00  1.123e+00  -2.444
## as.factor(district)160                    -7.712e-02  8.752e-01  -0.088
## living.children:as.factor(district)102  3.921e-01  5.219e-01   0.751
## living.children:as.factor(district)103 -1.064e-01  1.131e+03   0.000
## living.children:as.factor(district)104  1.682e-01  3.818e-01   0.441
## living.children:as.factor(district)105 -8.259e-03  3.136e-01  -0.026
## living.children:as.factor(district)106  5.898e-02  2.907e-01   0.203
## living.children:as.factor(district)107 -2.435e-01  4.590e-01  -0.530
## living.children:as.factor(district)108  4.354e-01  3.553e-01   1.226
## living.children:as.factor(district)109  1.643e-01  4.932e-01   0.333
## living.children:as.factor(district)110 -3.759e-01  8.876e-01  -0.423
## living.children:as.factor(district)111 -1.064e-01  4.978e+02   0.000
## living.children:as.factor(district)112 -4.454e-01  3.781e-01  -1.178
## living.children:as.factor(district)113  4.752e-01  3.854e-01   1.233
## living.children:as.factor(district)114  1.390e-01  2.398e-01   0.580
## living.children:as.factor(district)115  4.863e-01  4.400e-01   1.105
## living.children:as.factor(district)116  3.672e-01  4.861e-01   0.755
## living.children:as.factor(district)117  1.137e+00  8.029e-01   1.416
## living.children:as.factor(district)118 -1.182e-03  3.007e-01  -0.004
## living.children:as.factor(district)119  3.473e-01  3.719e-01   0.934
## living.children:as.factor(district)120 -1.064e-01  4.440e-01  -0.240
## living.children:as.factor(district)121 -1.273e+00  6.044e-01  -2.107
## living.children:as.factor(district)122 -5.143e-01  5.379e-01  -0.956
## living.children:as.factor(district)123  6.937e-01  7.544e-01   0.919
## living.children:as.factor(district)124  6.423e+00  3.581e+02   0.018
## living.children:as.factor(district)125  7.721e-01  3.055e-01   2.528
## living.children:as.factor(district)126  1.768e-01  6.419e-01   0.275
## living.children:as.factor(district)127 -9.857e-02  3.441e-01  -0.286
## living.children:as.factor(district)128  4.329e-01  3.711e-01   1.167
## living.children:as.factor(district)129  4.815e-01  3.676e-01   1.310
## living.children:as.factor(district)130  7.708e-01  3.141e-01   2.454
## living.children:as.factor(district)131  8.525e-02  3.264e-01   0.261
## living.children:as.factor(district)132  9.078e-02  5.088e-01   0.178
## living.children:as.factor(district)133  1.562e+00  9.376e-01   1.666
## living.children:as.factor(district)134 -3.898e-02  3.668e-01  -0.106
## living.children:as.factor(district)135  8.878e-02  3.140e-01   0.283
## living.children:as.factor(district)136 -9.241e-02  5.196e-01  -0.178
## living.children:as.factor(district)137 -3.632e-01  5.464e-01  -0.665
## living.children:as.factor(district)138  7.032e+00  4.666e+02   0.015
## living.children:as.factor(district)139 -4.006e-01  3.455e-01  -1.159
## living.children:as.factor(district)140 -1.337e-01  2.934e-01  -0.456
## living.children:as.factor(district)141 -4.390e-02  3.971e-01  -0.111
## living.children:as.factor(district)142  1.530e+01  7.770e+02   0.020
## living.children:as.factor(district)143  3.216e-01  3.208e-01   1.003
## living.children:as.factor(district)144  2.611e-01  4.432e-01   0.589
## living.children:as.factor(district)145  3.604e-01  3.856e-01   0.935
## living.children:as.factor(district)146  5.585e-02  2.552e-01   0.219
```

```
## living.children:as.factor(district)147  6.656e-01  5.079e-01   1.311
## living.children:as.factor(district)148  1.566e-01  3.273e-01   0.478
## living.children:as.factor(district)149 -1.064e-01  9.236e+02   0.000
## living.children:as.factor(district)150  6.251e-01  4.193e-01   1.491
## living.children:as.factor(district)151 -2.084e-01  3.315e-01  -0.629
## living.children:as.factor(district)152  2.385e-01  2.778e-01   0.858
## living.children:as.factor(district)153  6.511e-01  4.766e-01   1.366
## living.children:as.factor(district)154 -2.833e+01  8.508e+02  -0.033
## living.children:as.factor(district)155 -9.604e-02  3.012e-01  -0.319
## living.children:as.factor(district)156  4.303e-01  4.913e-01   0.876
## living.children:as.factor(district)157  6.988e-02  3.380e-01   0.207
## living.children:as.factor(district)158  6.310e-02  9.586e-01   0.066
## living.children:as.factor(district)159  4.673e-01  4.087e-01   1.143
## living.children:as.factor(district)160 -5.854e-01  3.774e-01  -1.551
##                                        Pr(>|z|)
## living.children                        0.555935
## as.factor(district)101                 0.016860 *
## as.factor(district)102                 0.181504
## as.factor(district)103                 0.995992
## as.factor(district)104                 0.444108
## as.factor(district)105                 0.287924
## as.factor(district)106                 0.054976 .
## as.factor(district)107                 0.599294
## as.factor(district)108                 0.033955 *
## as.factor(district)109                 0.276672
## as.factor(district)110                 0.356184
## as.factor(district)111                 0.987314
## as.factor(district)112                 0.700071
## as.factor(district)113                 0.070314 .
## as.factor(district)114                 0.864837
## as.factor(district)115                 0.073477 .
## as.factor(district)116                 0.462796
## as.factor(district)117                 0.085283 .
## as.factor(district)118                 0.178399
## as.factor(district)119                 0.089026 .
## as.factor(district)120                 0.708156
## as.factor(district)121                 0.100650
## as.factor(district)122                 0.855723
## as.factor(district)123                 0.164503
## as.factor(district)124                 0.984454
## as.factor(district)125                 0.000507 ***
## as.factor(district)126                 0.494618
## as.factor(district)127                 0.076915 .
## as.factor(district)128                 0.012810 *
## as.factor(district)129                 0.012544 *
## as.factor(district)130                 0.001486 **
## as.factor(district)131                 0.393789
## as.factor(district)132                 0.236723
## as.factor(district)133                 0.071295 .
## as.factor(district)134                 0.653708
## as.factor(district)135                 0.481655
## as.factor(district)136                 0.660177
## as.factor(district)137                 0.576785
## as.factor(district)138                 0.987620
```

```
## as.factor(district)139                0.377492
## as.factor(district)140                0.914651
## as.factor(district)141                0.869544
## as.factor(district)142                0.984177
## as.factor(district)143                0.208964
## as.factor(district)144                0.073251 .
## as.factor(district)145                0.052798 .
## as.factor(district)146                0.515140
## as.factor(district)147                0.125532
## as.factor(district)148                0.460714
## as.factor(district)149                0.993434
## as.factor(district)150                0.078245 .
## as.factor(district)151                0.885215
## as.factor(district)152                0.066767 .
## as.factor(district)153                0.077388 .
## as.factor(district)154                0.976687
## as.factor(district)155                0.655107
## as.factor(district)156                0.050741 .
## as.factor(district)157                0.435114
## as.factor(district)158                0.369937
## as.factor(district)159                0.014538 *
## as.factor(district)160                0.929781
## living.children:as.factor(district)102 0.452516
## living.children:as.factor(district)103 0.999925
## living.children:as.factor(district)104 0.659553
## living.children:as.factor(district)105 0.978988
## living.children:as.factor(district)106 0.839229
## living.children:as.factor(district)107 0.595840
## living.children:as.factor(district)108 0.220344
## living.children:as.factor(district)109 0.738961
## living.children:as.factor(district)110 0.671964
## living.children:as.factor(district)111 0.999830
## living.children:as.factor(district)112 0.238764
## living.children:as.factor(district)113 0.217534
## living.children:as.factor(district)114 0.562115
## living.children:as.factor(district)115 0.269072
## living.children:as.factor(district)116 0.450067
## living.children:as.factor(district)117 0.156763
## living.children:as.factor(district)118 0.996863
## living.children:as.factor(district)119 0.350370
## living.children:as.factor(district)120 0.810650
## living.children:as.factor(district)121 0.035154 *
## living.children:as.factor(district)122 0.339033
## living.children:as.factor(district)123 0.357841
## living.children:as.factor(district)124 0.985689
## living.children:as.factor(district)125 0.011487 *
## living.children:as.factor(district)126 0.782947
## living.children:as.factor(district)127 0.774529
## living.children:as.factor(district)128 0.243321
## living.children:as.factor(district)129 0.190237
## living.children:as.factor(district)130 0.014136 *
## living.children:as.factor(district)131 0.793987
## living.children:as.factor(district)132 0.858387
## living.children:as.factor(district)133 0.095626 .
```

```
## living.children:as.factor(district)134 0.915366
## living.children:as.factor(district)135 0.777354
## living.children:as.factor(district)136 0.858838
## living.children:as.factor(district)137 0.506233
## living.children:as.factor(district)138 0.987975
## living.children:as.factor(district)139 0.246355
## living.children:as.factor(district)140 0.648495
## living.children:as.factor(district)141 0.911982
## living.children:as.factor(district)142 0.984287
## living.children:as.factor(district)143 0.316094
## living.children:as.factor(district)144 0.555693
## living.children:as.factor(district)145 0.349941
## living.children:as.factor(district)146 0.826738
## living.children:as.factor(district)147 0.189990
## living.children:as.factor(district)148 0.632365
## living.children:as.factor(district)149 0.999908
## living.children:as.factor(district)150 0.136040
## living.children:as.factor(district)151 0.529483
## living.children:as.factor(district)152 0.390667
## living.children:as.factor(district)153 0.171859
## living.children:as.factor(district)154 0.973437
## living.children:as.factor(district)155 0.749852
## living.children:as.factor(district)156 0.381028
## living.children:as.factor(district)157 0.836232
## living.children:as.factor(district)158 0.947515
## living.children:as.factor(district)159 0.252905
## living.children:as.factor(district)160 0.120881
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2681.1  on 1934  degrees of freedom
## Residual deviance: 2285.1  on 1814  degrees of freedom
## AIC: 2525.1
##
## Number of Fisher Scoring iterations: 15
```

This model formula creates an interaction term between each district and living children, and so each coefficient of the interaction terms represents an individual model for each district since the -1 ensures there is no intercept. It looks like the sign and magnitude of the coefficient for each district differs (though not all statistically significant), indicating that living children have different impacts on contraceptive use depending on the individual district.

1(c) Bayesian Hierarchical Logistic Model

```r
model.hierarchical <- MCMChlogit(fixed = contraceptive_use ~ living.children, random = ~ living.childre
```

```
##
## Running the Gibbs sampler. It may be long, keep cool :)
##
## **********:10.0%, mean accept. rate=0.422
## **********:20.0%, mean accept. rate=0.469
```

```
## **********:30.0%, mean accept. rate=0.409
## **********:40.0%, mean accept. rate=0.453
## **********:50.0%, mean accept. rate=0.415
## **********:60.0%, mean accept. rate=0.461
## **********:70.0%, mean accept. rate=0.523
## **********:80.0%, mean accept. rate=0.449
## **********:90.0%, mean accept. rate=0.523
## **********:100.0%, mean accept. rate=0.456
```

```
summary(model.hierarchical)
```

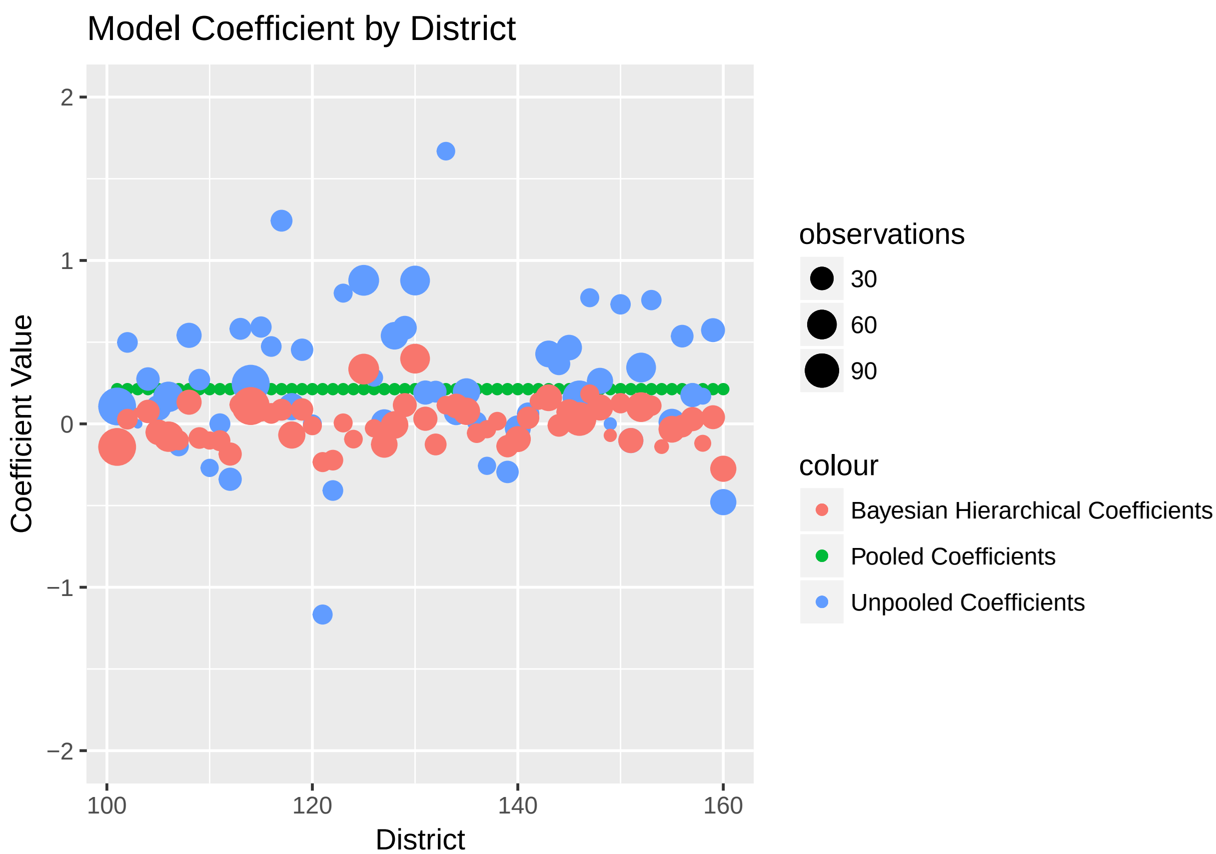
```
##             Length Class  Mode
## mcmc        12800  mcmc   numeric
## theta.pred  1934   -none- numeric
```

The model returns distributions of intercepts and living children coefficients for each district as well as the estimated probability for each district model. The pooled and unpooled model do not these return probabilities.

2(a) Plot living.children Coefficients

```
# Get list and number of districts
districts <- unique(dataset_2$district)
n_districts <- length(districts)

# Get pooled coefficients
pooled_coefficients <- rep(model.pooled$coefficients[2], n_districts)

# Get unpooled coeffficents
all_unpooled_coefficients <- model.unpooled$coefficient
n_all_unpooled_coefficients <- length(all_unpooled_coefficients)
unpooled_coefficients <- all_unpooled_coefficients[(n_all_unpooled_coefficients - n_districts + 2):n_all
unpooled_coefficients <- c(0, unpooled_coefficients) + all_unpooled_coefficients[1]

# Get bayesian hierarchical model coefficients
all_hierarchical_coefficients <- summary(model.hierarchical$mcmc)$statistics[, 1]
n_all_hierarchical_coefficients <- length(all_hierarchical_coefficients)
hierarchical_coefficients <- all_hierarchical_coefficients[(n_all_hierarchical_coefficients - 5 - n_dist

# Plot coefficients
coefficient_graph_data <- data.frame(districts = districts, observations = as.numeric(table(dataset_2$di
ggplot(coefficient_graph_data, aes(districts)) + geom_point(aes(y = pooled, colour = "Pooled Coefficient
```

```
## Warning: Removed 4 rows containing missing values (geom_point).
```



Vertical axis set between -2 and 2, since any coefficients larger than those that have little practical significance and eliminating them increases visibility into the variance in coefficients in the Bayesian hierarchical model.

2(b) Plot Summary

Both the Bayesian hierarchical coefficients and the unpooled coefficients exhibit some variance depending on the district, however the unpooled coefficients appear to have a much higher variance in terms of coefficient value. The Bayesian hierarchical coefficients appear to be much more tightly clustered around the pooled coefficient as compared to the unpooled coefficients. In addition, only 2 districts have Bayesian hierarchical coefficients that are greater than the pooled coefficient, indicating it may act as a sort of upper-bound for most districts. The Bayesian hierarchical coefficients therefore look to be a compromise between the unpooled and pooled coefficients, allowing for variability but still accounting for the overall trend in the dataset.

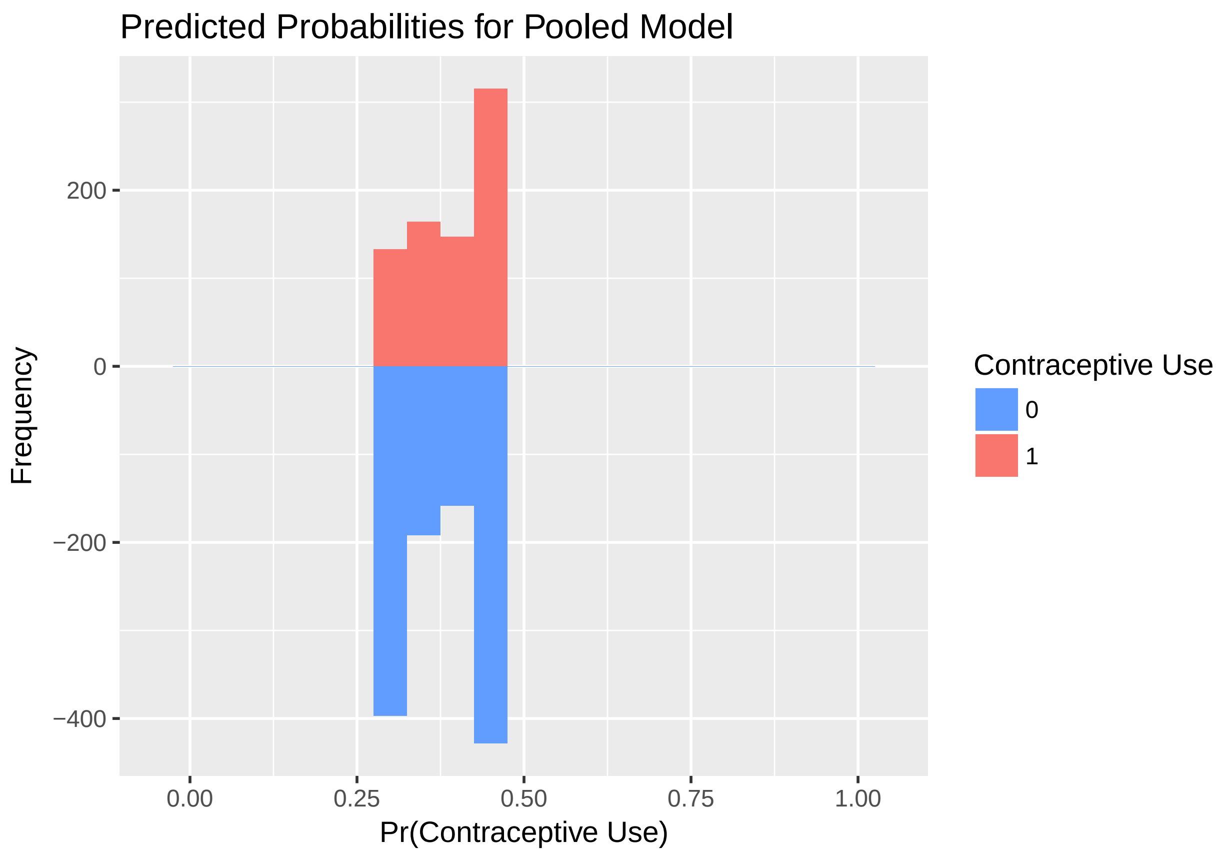
The number of observations in a particular district appears to effect the variability of the unpooled and Bayesian hierarchical coefficients, as those with a lower number of observations tend to have coefficient values further away from the pooled coefficient. This could indicate potential over-fitting due to small sample size.

3(a) Model Histograms

Pooled Model

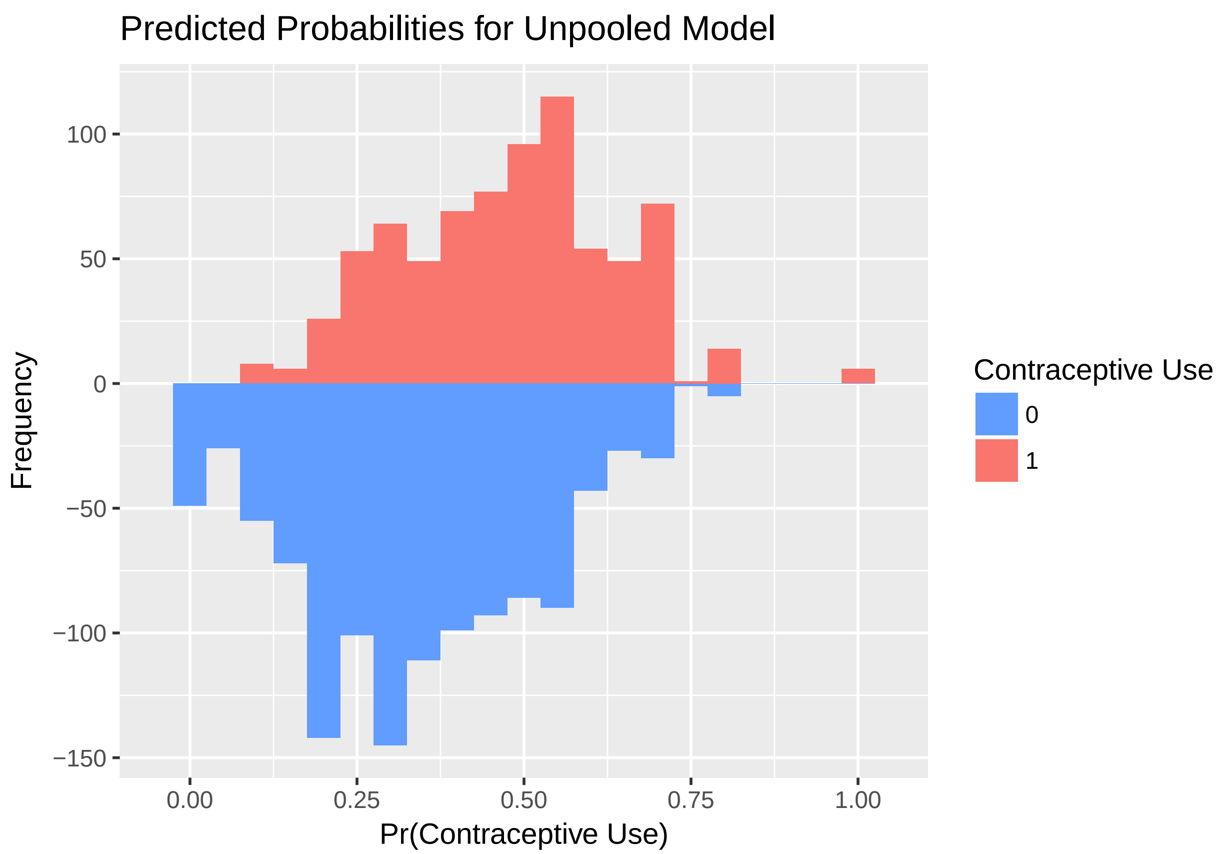
```
# Calculate and standardize pooled probabilties
pred.pooled <- predict(model.pooled, dataset_2, type = "response")
dataset_2$pooled_prob <- pred.pooled
```

```
# Plot probabilities
binwidth <- .05
ggplot() + geom_histogram(data = dataset_2[dataset_2$contraceptive_use == 1, ], aes(x = pooled_prob, fil
```



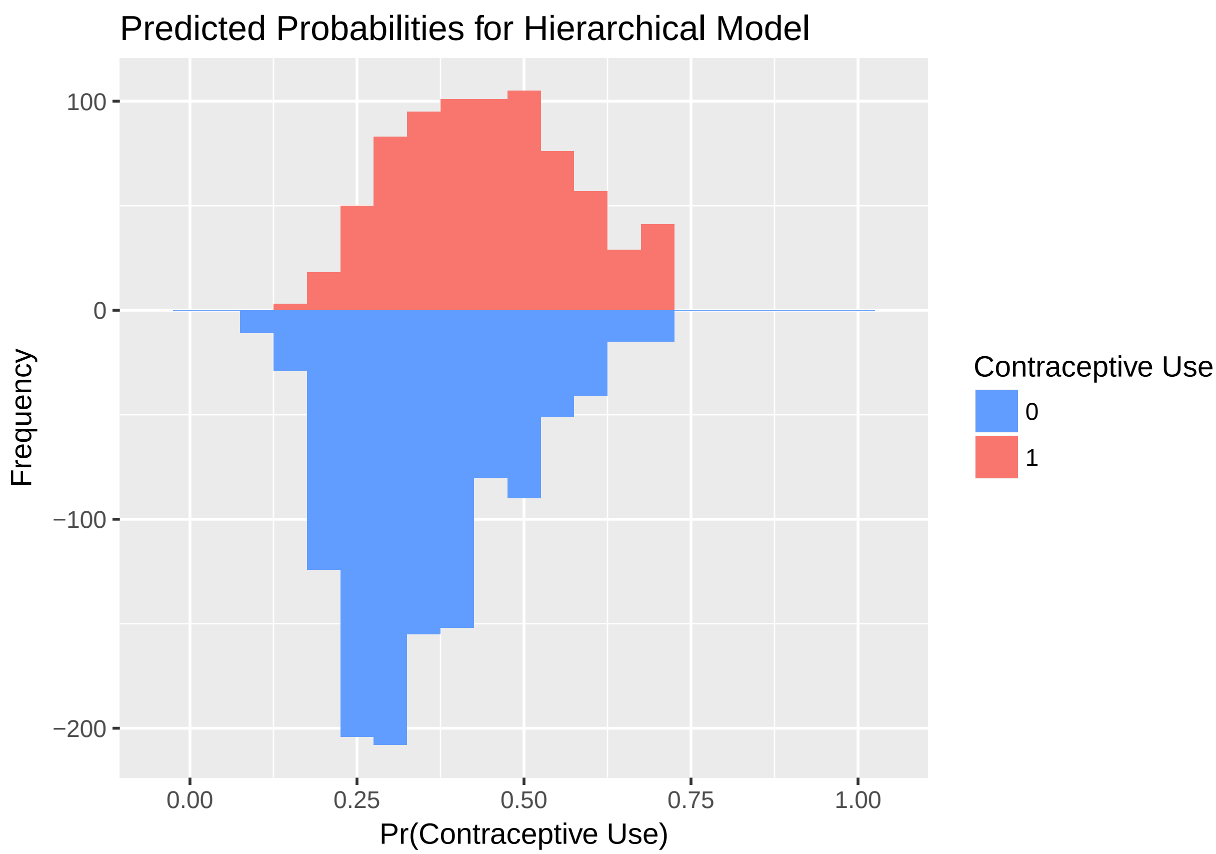
Unpooled Model

```
# Calculate and standardize unpooled probabilties
pred.unpooled <- predict(model.unpooled, dataset_2, type = "response")
dataset_2$unpooled_prob <- pred.unpooled

# Plot probabilities
binwidth <- .05
ggplot() + geom_histogram(data = dataset_2[dataset_2$contraceptive_use == 1, ], aes(x = unpooled_prob, f
```



Hierarchical Model

```
# Calculate and standardize pooled probabilties
pred.hierarchical <- model.hierarchical$theta.pred
dataset_2$hierarchical_prob <- pred.hierarchical

# Plot probabilities
binwidth <- .05
ggplot() + geom_histogram(data = dataset_2[dataset_2$contraceptive_use == 1, ], aes(x = hierarchical_pr
```

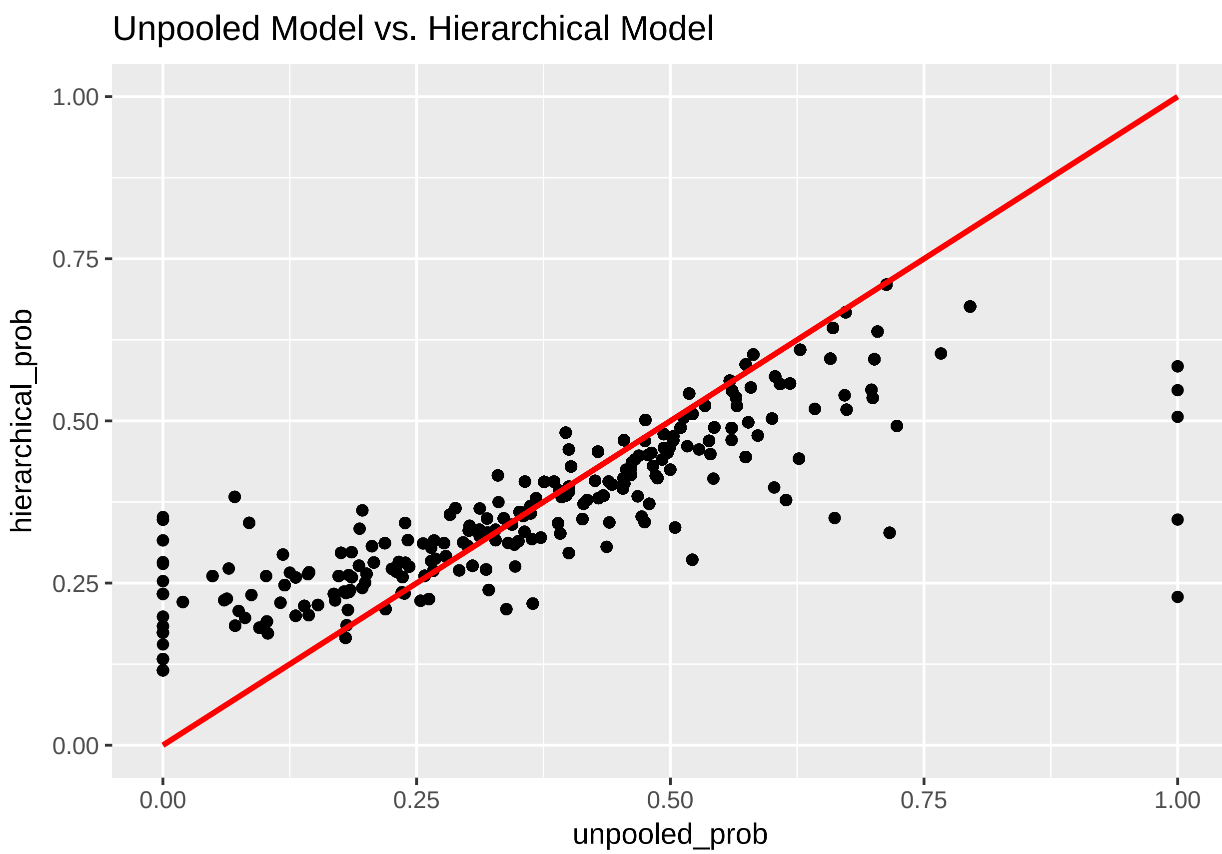


Model Comparison

The pooled model histogram exhibits probabilities for all 4 potential values of living children (with probability increasing as number of children does). It clearly does a poor job of distinguishing between classes as the no contraceptive use dominates regardless of the probability threshold chosen. The unpooled model histogram probabilities are more spread out (though still clustered between .25 and .75) and it does a better job differentiating between the classes, particularly at probability values further away from .5. The hierarchical model histogram is more clustered than the unpooled model and appears to differentiate between the classes better around the .5 level than the unpooled model (though neither does particularly well).

3(b) Unpooled Hierarchical Comparison

```
ggplot(data = dataset_2, aes(x = unpooled_prob)) + geom_point(aes(y = hierarchical_prob)) + geom_line(a
```



The probabilities of both models look to be positively correlated, however the hierarchical model probabilities appear to be less 'confident' (further away from 0 or 1, closer to .5 than the unpooled model probabilities). This makes sense as the hierarchical model is incorporating the 'prior' of the pooled model, which helps prevent it from over-fitting to particular district's data.

Problem 3

- **Email Address**: ihsaan.patel@gmail.com
- **Harvard Email**: ihsaan_patel@hks18.harvard.edu
- **AWS ID**: 157598045262