Problem_Set_6

Carlos Sathler

4/10/2019

Contents

Problem 1

Can I predict number of colonies from dose level? * Dataset: Salmonella in faraway library * Response variable: number of colonies * Predictor: dose level o quinoline

EDA

```
df = salmonella
str(salmonella)
```

```
## 'data.frame':    18 obs. of  2 variables:
##  $ colonies: int  15 21 29 16 18 21 16 26 33 27 ...
##  $ dose    : int  0 0 0 10 10 10 33 33 33 100 ...
```

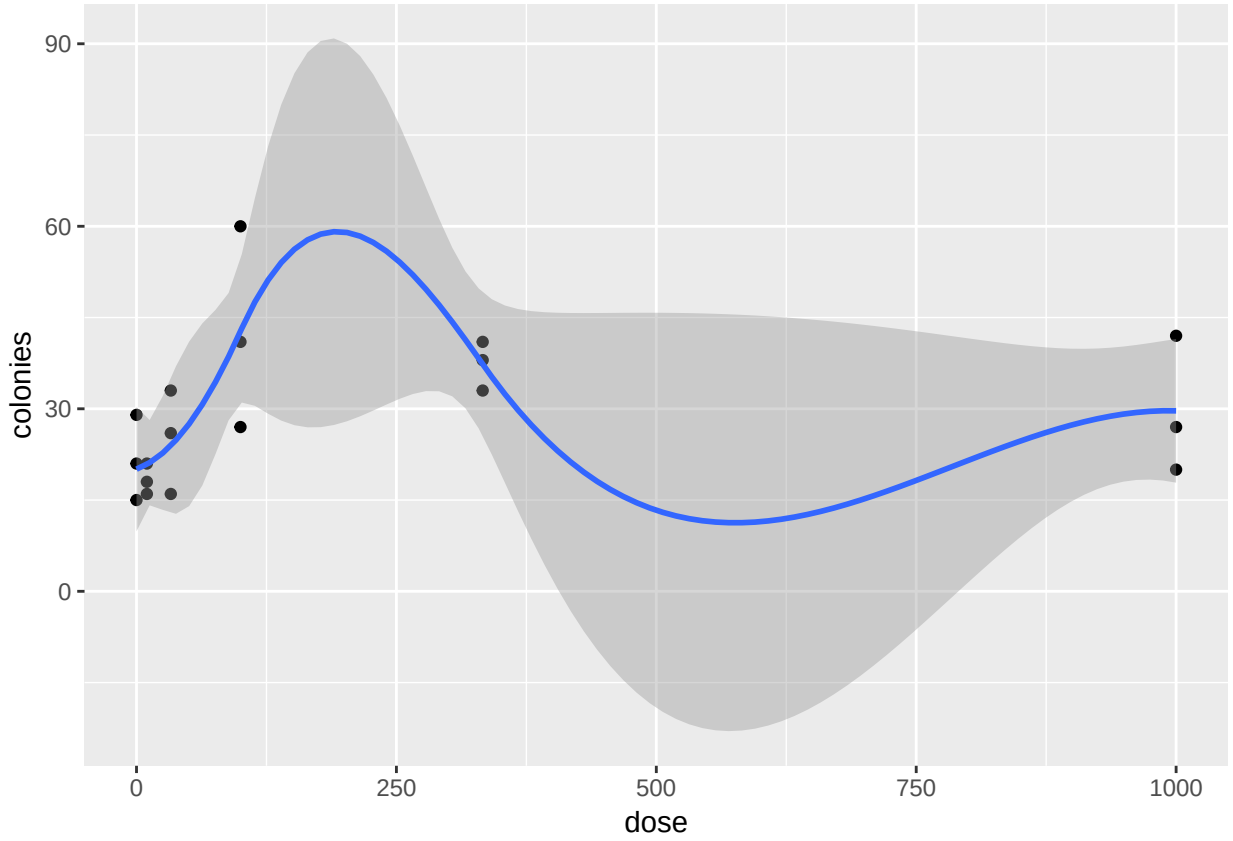
```
summary(salmonella)
```

```
##     colonies          dose
##  Min.   :15.00   Min.   :   0.0
##  1st Qu.:20.25   1st Qu.:  10.0
##  Median :27.00   Median :  66.5
##  Mean   :29.11   Mean   : 246.0
##  3rd Qu.:36.75   3rd Qu.: 333.0
##  Max.   :60.00   Max.   :1000.0
```

```
y = df$colonies
x = df$dose
cor(x, y)
```

```
## [1] 0.1783561
```

```
ggplot(df, aes(x=dose, y=colonies)) + geom_point() + geom_smooth(method='loess')
```



Model

After some online research (http://www.nib.si/eng/index.php/services-and-products?id=56) I discovered that there is an expectation that the number of revertant colonies should increase with the increase of dose. I therefore will start with a poisson regression model since, per class lecture, "the Poisson distribution is often applicable when the expected count is proportional to some quantity or quantities". Another way to consider the problem is that we are looking for how many colony "reversions" happen per dose (λ), in other words, I can model this problem as a Poisson process, i.e., a counting or "arrival" problem.

```
# I use glm function in mgcv package with family set to poisson
library(mgcv)
```

```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
## This is mgcv 1.8-26. For overview type 'help("mgcv-package")'.
```

```
salm.glm = glm(colonies ~ dose, family = poisson, data = salmonella)
tidy(salm.glm)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept) 3.32      0.0540       61.5    0
## 2 dose        0.000190  0.000117      1.62   0.105
```

According to class notes, "in a standard Poisson regression, the response has a Poisson distribution with the log of the expected value given by a linear function of the predictors. In the single-variable case: $\log(E[Y|x]) = \beta_0 + \beta_1 x$"

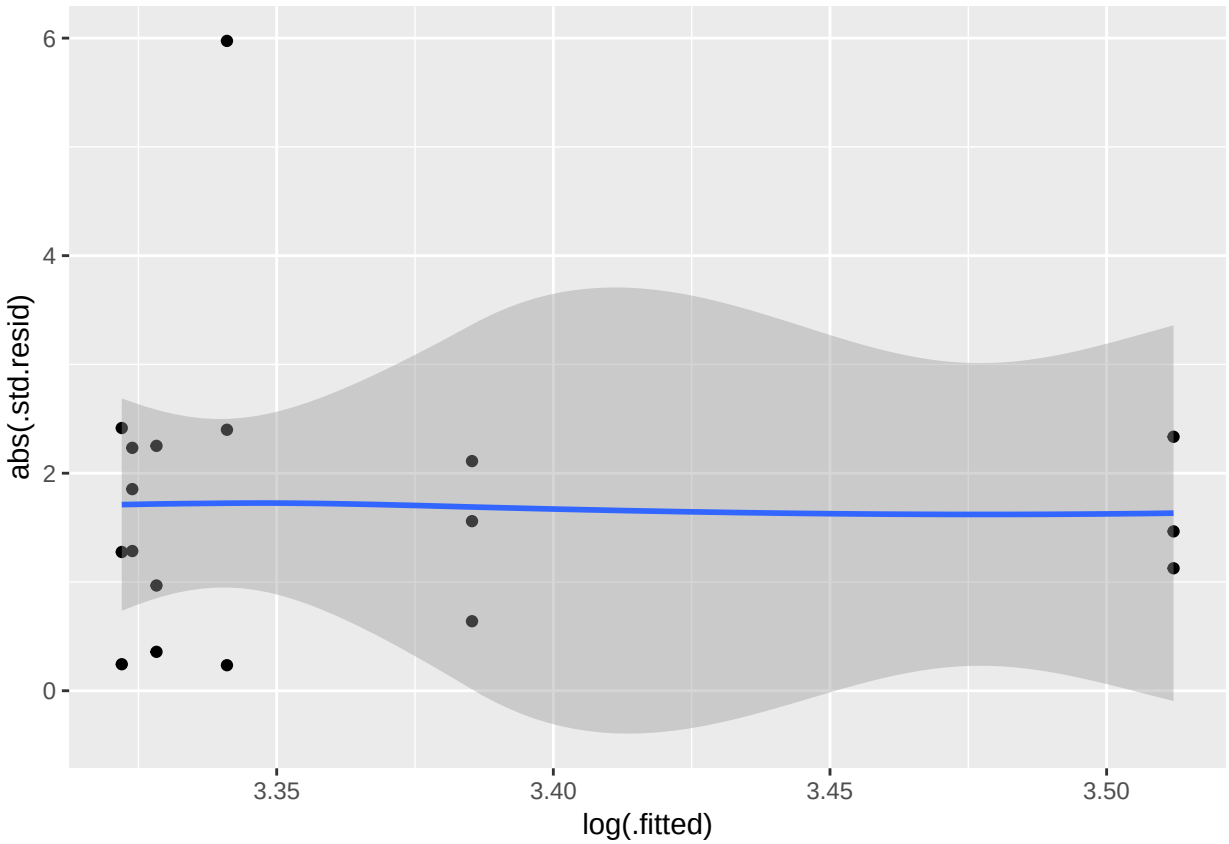
For this problem: $log(colonies) = 3.322 + 0.0002 * dose$

We note that the coefficient for dose is not much larger than std.error, and p.value is not close to zero, suggesting dose may not be relevant to reverse mutagencity of TA98 Salmonella. That is not surprising given the low correlation between the variables and the shape of the loess smoother in the plot of reversals vs. dose above.

Next I check residuals for the model.

```
# get fitted values, which function returns in real prediction value of number of colonies
salm.glm.fitted = fitted.values(salm.glm)
# get residuals, which would be returned as log(residuals) without type='response'
salm.glm.resid = residuals(salm.glm, type = "response")
# calculate standardized residulas for plotting
salm.glm.std.resid = salm.glm.resid / sqrt(salm.glm.fitted)
salm.glm.df = data.frame(df, .fitted = salm.glm.fitted, .std.resid = salm.glm.std.resid)
gg = ggplot(salm.glm.df, aes(x = log(.fitted), y = abs(.std.resid))) + geom_point()
# using loess with maximum smoothing and degree one smoothing polynomial
gg + geom_smooth(span = 1, method.args = list(degree = 1))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Distribution of residuals is not too bad and I will consider it reasonably homoskedastic. Still, according to class, Poisson regression is commonly probabilistically wrong, i.e., while it often predicts well, it can be overly dispersed. With that in mind, I'll check for overdispersion.

```
sum(salm.glm.std.resid^2) / df.residual(salm.glm)
```

```
## [1] 5.087258
```

The residuals are overdispersed ($5.09 > 2$).

I will address the issue by fitting a dispersed Poisson model with the quasipoisson family in glm.

```
salm.glm.q = glm(colonies ~ dose, family = quasipoisson, data = salmonella)
tidy(salm.glm.q)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept) 3.32      0.122       27.3   7.72e-15
## 2 dose        0.000190  0.000264     0.719 4.82e- 1
```

Standard errors are much higher than before, as expected. It is still questionable if dose really matters.

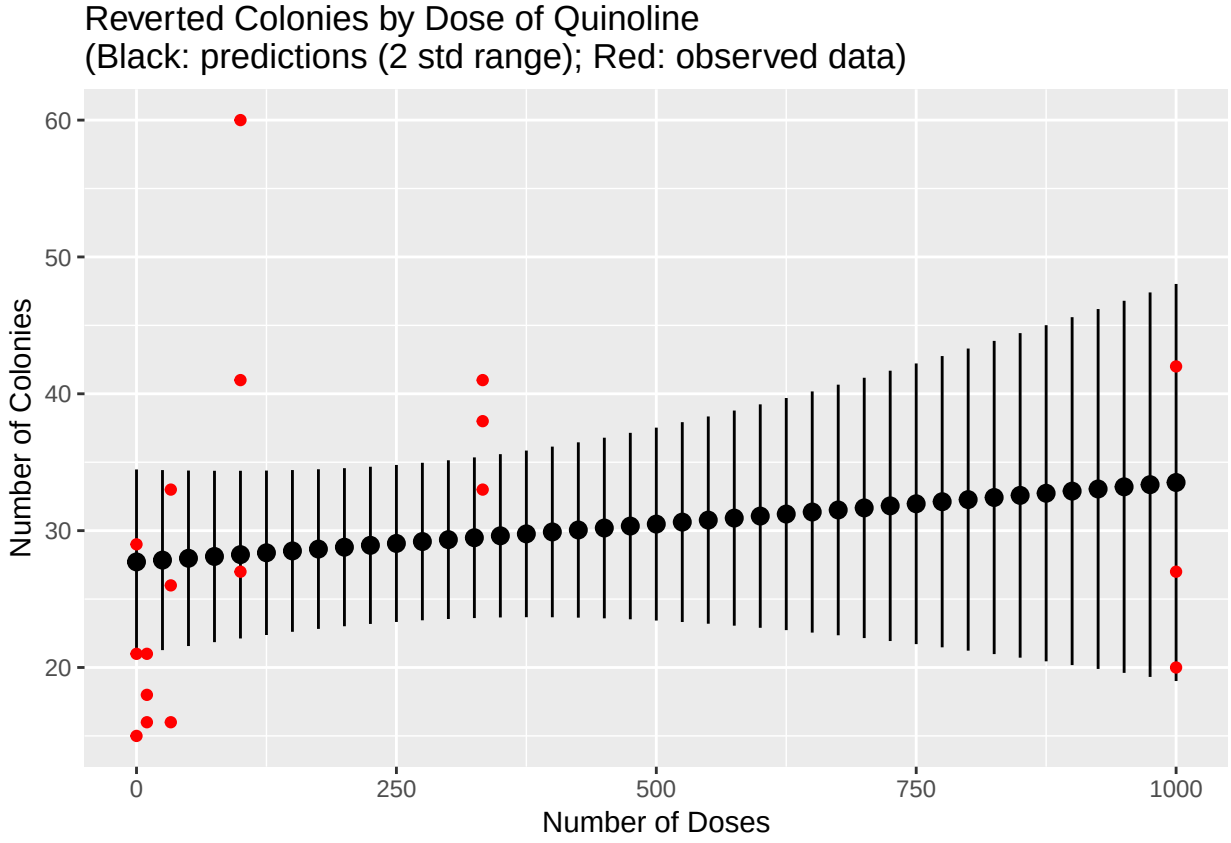
Next, I plot my final model, with standard errors for predictions. Red points are observations.

```
# predicts 200 values of colonies, and get standard error for predictions
doses.df = data.frame(dose=seq(0, 1000, 25))
pred = predict(salm.glm.q, newdata = doses.df, se.fit=T, type='response')
# get estimates alone
estimate = pred$fit
# find lower and upper intervals for estimates
```

```
lower = pred$fit - 2 * pred$se.fit
upper = pred$fit + 2 * pred$se.fit

col.est.df = data.frame(doses.df, estimate, lower, upper)
gg = ggplot(col.est.df, aes(x = dose, y = estimate, ymin = lower, ymax = upper)) + geom_pointrange()
gg = gg + ylab('Number of Colonies') + xlab('Number of Doses') + ylim(15,60)
gg = gg + geom_point(data=df, mapping = aes(x=dose, y=colonies, ymin=1, ymax=10), color = 'red')
```

```
## Warning: Ignoring unknown aesthetics: ymin, ymax
```

```
gg = gg + ggtitle('Reverted Colonies by Dose of Quinoline\n(Black: predictions (2 std range); Red: obse
gg
```



Conclusion

The model leaves many points outside of prediction confidence bands (2 standard deviations), especially for lower values of dose. I looked online and Quinoline doesn't seem to be a particularly expensive compound. It would be interesting to see how the model performs for number of doses greater than 375, and particularly for doses greater than 1,000.

Problem 2

Can I predict whether tumor is malign or benign using a number of explanatory variables from the Wisconsin breast cancer database? * Dataset: wbca from faraway library * Response: Class (0 for begning tumor; 1 for malign) * Predictors: 9 variables on a 1 (normal) to 10 (abnormal) scale

EDA

EDA will show most predictors are skewed to the normal side of the scale. The proportion of maligned vs. benign tumors is 1:2

```
# Class is response variable: it is 0 if the tumor is malignant and 1 if the tumor is benign.
# The other variables are predictors on a 1 (normal) to 10 (abnormal) scale, as rated by a doctor.
train = read.table('wbca-training.txt', header=T)
test = read.table('wbca-test.txt', header=T)
str(train)
```

```
## 'data.frame':    481 obs. of  10 variables:
##  $ Class: int  1 1 1 0 1 1 1 1 0 1 ...
##  $ Adhes: int  1 5 1 8 1 1 1 1 3 1 ...
##  $ BNucl: int  1 10 4 10 10 1 1 1 3 3 ...
##  $ Chrom: int  3 3 3 9 3 3 1 2 4 3 ...
##  $ Epith: int  2 7 3 7 2 2 2 2 2 2 ...
##  $ Mitos: int  1 1 1 1 1 1 5 1 1 1 ...
##  $ NNucl: int  1 2 7 7 1 1 1 1 4 1 ...
##  $ Thick: int  5 5 6 8 1 2 2 4 5 1 ...
##  $ UShap: int  1 4 8 10 1 2 1 1 3 1 ...
##  $ USize: int  1 4 8 10 1 1 1 2 3 1 ...
```

```
summary(train)
```

```
##      Class            Adhes            BNucl            Chrom       
##  Min.   :0.0000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000  
##  1st Qu.:0.0000   1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.: 2.000  
##  Median :1.0000   Median : 1.000   Median : 1.000   Median : 3.000  
##  Mean   :0.6611   Mean   : 2.753   Mean   : 3.493   Mean   : 3.372  
##  3rd Qu.:1.0000   3rd Qu.: 3.000   3rd Qu.: 5.000   3rd Qu.: 4.000  
##  Max.   :1.0000   Max.   :10.000   Max.   :10.000   Max.   :10.000  
##      Epith            Mitos            NNucl           Thick       
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.00   Min.   : 1.000  
##  1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.00   1st Qu.: 2.000  
##  Median : 2.000   Median : 1.000   Median : 1.00   Median : 4.000  
##  Mean   : 3.266   Mean   : 1.628   Mean   : 2.84   Mean   : 4.416  
##  3rd Qu.: 4.000   3rd Qu.: 1.000   3rd Qu.: 3.00   3rd Qu.: 6.000  
##  Max.   :10.000   Max.   :10.000   Max.   :10.00   Max.   :10.000  
##      UShap            USize       
##  Min.   : 1.000   Min.   : 1.000  
##  1st Qu.: 1.000   1st Qu.: 1.000  
##  Median : 1.000   Median : 1.000  
##  Mean   : 3.187   Mean   : 3.112  
##  3rd Qu.: 5.000   3rd Qu.: 5.000  
##  Max.   :10.000   Max.   :10.000
```

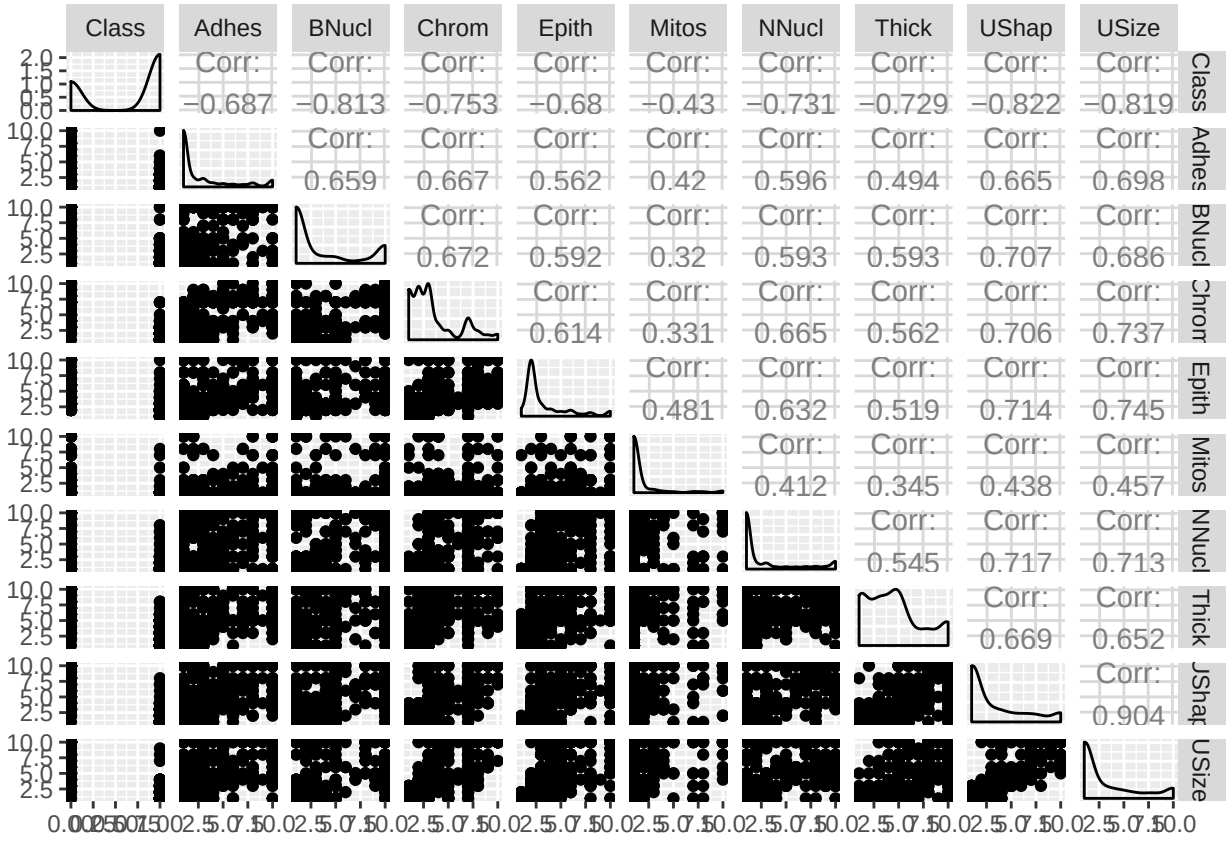
```
print('Proportion of malign vs. begning tumors')
```

```
## [1] "Proportion of malign vs. begning tumors"
```

```
summary(factor(train$Class))
```

```
##   0   1
## 163 318
```

```
# plot correlations
ggpairs(train, progress = FALSE)
```



(a) logistic regression

Here I'll create a model with all predictors and will run predictions on test dataset. I will then print confusion matrix and check accuracy.

```
# train model using all variables as predictors
model.a.logit = glm(Class ~ ., family = "binomial", data = train)
tidy(model.a.logit)
```

```
## # A tibble: 10 x 5
##    term        estimate std.error statistic  p.value
##    <chr>          <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)   11.7       1.84      6.36  2.00e-10
##  2 Adhes         -0.310     0.155    -2.01  4.49e- 2
##  3 BNucl         -0.434     0.120    -3.63  2.82e- 4
##  4 Chrom         -0.734     0.219    -3.35  8.08e- 4
##  5 Epith          0.148     0.205     0.724 4.69e- 1
##  6 Mitos         -0.472     0.395    -1.19  2.33e- 1
```

```
##  7 NNucl         -0.380     0.159    -2.39  1.68e- 2
##  8 Thick         -0.734     0.207    -3.55  3.88e- 4
##  9 UShap         -0.307     0.294    -1.04  2.96e- 1
## 10 USize          0.127     0.270     0.471 6.38e- 1
```

```
# test model on test dataset
test2 = test[,-1]
preds = predict(model.a.logit, newdata = test2, type='response')
# per instructions: a tumor is predicted to be benign if the model probability is at least 0.5
# and malignant otherwise
preds.bin = round(preds, 0)
# get accuracy using confusion matrix function from caret package
conf = confusionMatrix(as.factor(preds.bin), as.factor(test$Class))
conf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  70   0
##          1   5 125
##
##                Accuracy : 0.975
##                  95% CI : (0.9426, 0.9918)
##     No Information Rate : 0.625
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9459
##
##  Mcnemar's Test P-Value : 0.07364
##
##             Sensitivity : 0.9333
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9615
##              Prevalence : 0.3750
##          Detection Rate : 0.3500
##    Detection Prevalence : 0.3500
##       Balanced Accuracy : 0.9667
##
##        'Positive' Class : 0
##
```

```
conf$overall[1]
```

```
## Accuracy
##    0.975
```

The proportion of observations predicted correctly is 97.5%.

(b) more sophisticated model

The model with all variables achieved pretty good accuracy (97.5%). I'll try to beat it with shrinkage, specifically ridge regression, lasso, and a third model using maximum regularization. These techniques require scaling the predictors, so I'll do that first.

Scale predictors

```
Class = train[,1]
train.scaled = train[,-1]
train.scaled = scale(train.scaled)
summary(train.scaled)
```

```
##      Adhes              BNucl              Chrom              Epith
##  Min.   :-0.62293   Min.   :-0.6954   Min.   :-0.9856   Min.   :-0.9938
##  1st Qu.:-0.62293   1st Qu.:-0.6954   1st Qu.:-0.5701   1st Qu.:-0.5552
##  Median :-0.62293   Median :-0.6954   Median :-0.1546   Median :-0.5552
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.08793   3rd Qu.: 0.4205   3rd Qu.: 0.2609   3rd Qu.: 0.3218
##  Max.   : 2.57594   Max.   : 1.8153   Max.   : 2.7539   Max.   : 2.9530
##      Mitos              NNucl              Thick              UShap
##  Min.   :-0.3441   Min.   :-0.6022   Min.   :-1.2116   Min.   :-0.7332
##  1st Qu.:-0.3441   1st Qu.:-0.6022   1st Qu.:-0.8569   1st Qu.:-0.7332
##  Median :-0.3441   Median :-0.6022   Median :-0.1475   Median :-0.7332
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.:-0.3441   3rd Qu.: 0.0524   3rd Qu.: 0.5619   3rd Qu.: 0.6077
##  Max.   : 4.5879   Max.   : 2.3435   Max.   : 1.9808   Max.   : 2.2839
##      USize
##  Min.   :-0.6861
##  1st Qu.:-0.6861
##  Median :-0.6861
##  Mean   : 0.0000
##  3rd Qu.: 0.6131
##  Max.   : 2.2371
```

```
# now use caret preprocess function to scale test dataset using training mean and sd values
scaleParm = preProcess(train[,-1])
test2.scaled = predict(scaleParm, test2)
```

Shrinkage with ridge regression

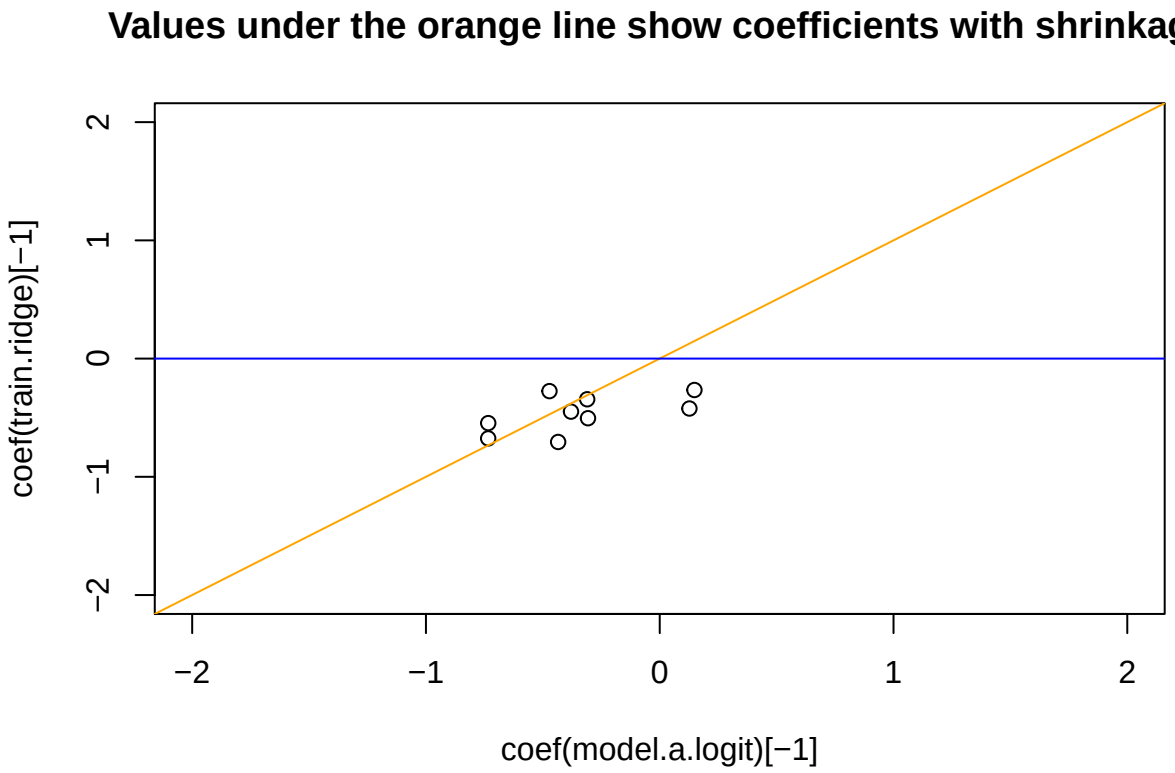
Create function to test model and return accuracy

```
get_accuracy = function(model, newdata) {
  preds = predict(model, newx = newdata, type='response')
  # per instructions: a tumor is predicted to be benign if the model probability is at least 0.5
  # and malignant otherwise
  preds.bin = round(preds, 0)
  # get accuracy using confusion matrix function from caret package
  conf = confusionMatrix(as.factor(preds.bin), as.factor(test$Class))
  return(conf$overall[1])
}
```

```
train.ridge.cv = cv.glmnet(train.scaled, Class, family = "binomial", alpha = 0)
train.ridge = glmnet(train.scaled, Class, alpha = 0, family = "binomial", lambda = train.ridge.cv$lambda
coef(train.ridge)
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
## (Intercept)  1.0532006
## Adhes       -0.3442676
## BNucl       -0.7053664
## Chrom       -0.5457951
## Epith       -0.2656158
## Mitos       -0.2751831
## NNucl       -0.4485565
## Thick       -0.6751208
## UShap       -0.5044548
## USize       -0.4221568
```

```
# confirm shrinkage by comparison with model from part a
main = 'Values under the orange line show coefficients with shrinkage'
plot(coef(model.a.logit)[-1], coef(train.ridge)[-1], ylim=c(-2,2), xlim=c(-2,2), main=main)
abline(0, 1, col = "orange")
abline(h = 0, col = "blue")
```




```
print('Accuracy from Ridge Regression:')
```

```
## [1] "Accuracy from Ridge Regression:"
```

```
get_accuracy(train.ridge, as.matrix(test2.scaled))
```

```
## Accuracy
##    0.975
```

Accuracy from ridge regression matches accuracy from logistic regression, therefore it offers no improvement over the first model.

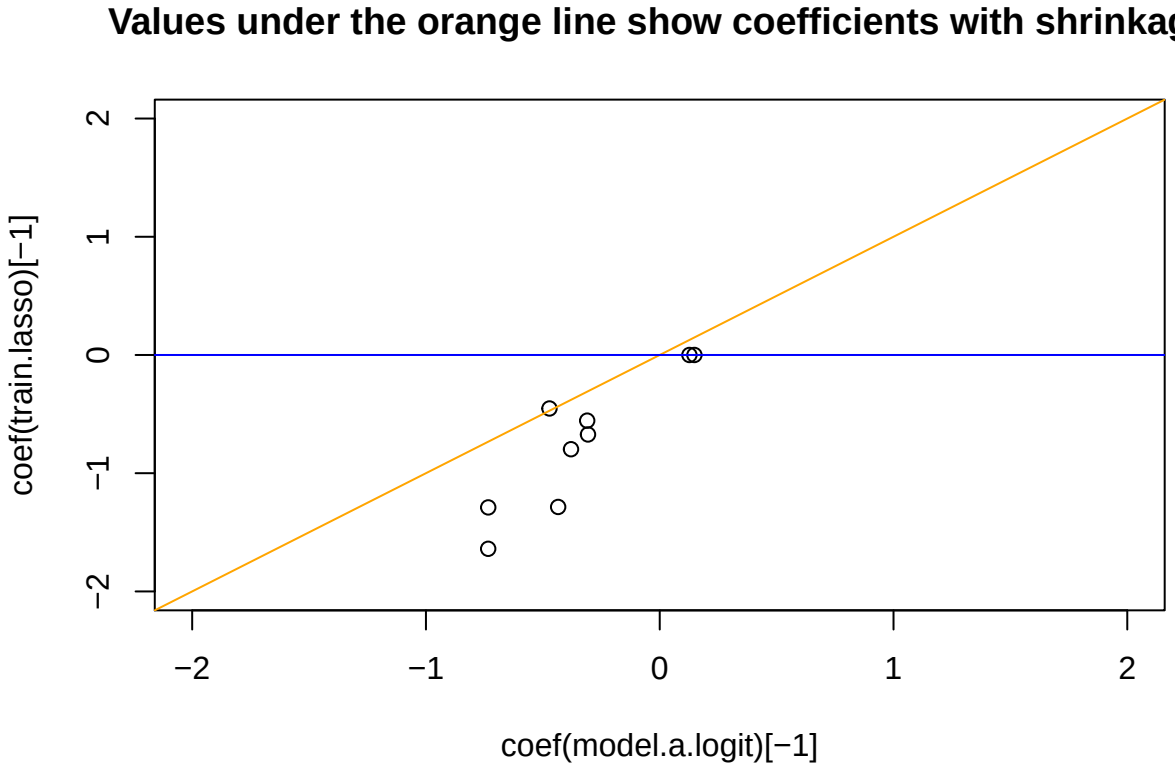
Shrinkage with lasso

Note that with lasso some coefficients become zero.

```
# for lasso, we repeat same steps used for ridge regression, but set alpha = 1
train.lasso.cv = cv.glmnet(train.scaled, Class, family = "binomial", alpha = 1)
train.lasso = glmnet(train.scaled, Class, alpha = 1, family = "binomial", lambda = train.lasso.cv$lambda
coef(train.lasso)
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
## (Intercept)  1.4759799
## Adhes       -0.5554229
## BNucl       -1.2855927
## Chrom       -1.2892859
## Epith        .
## Mitos       -0.4525152
## NNucl       -0.7970597
## Thick       -1.6393764
## UShap       -0.6722996
## USize        .
```

```
# confirm shrinkage by comparison with model from part a
main = 'Values under the orange line show coefficients with shrinkage'
plot(coef(model.a.logit)[-1], coef(train.lasso)[-1], ylim=c(-2,2), xlim=c(-2,2), main=main)
abline(0, 1, col = "orange")
abline(h = 0, col = "blue")
```



```
print('Accuracy from Lasso:')
```

```
## [1] "Accuracy from Lasso:"
```

```
get_accuracy(train.ridge, as.matrix(test2.scaled))
```

```
## Accuracy
##    0.975
```

Accuracy from lasso matches accuracy from logistic regression however, the model is simpler (Epith and USize predictors were dropped). Therefore if accuracy is the only metric to compare these models, the lasso is superior to the first model.

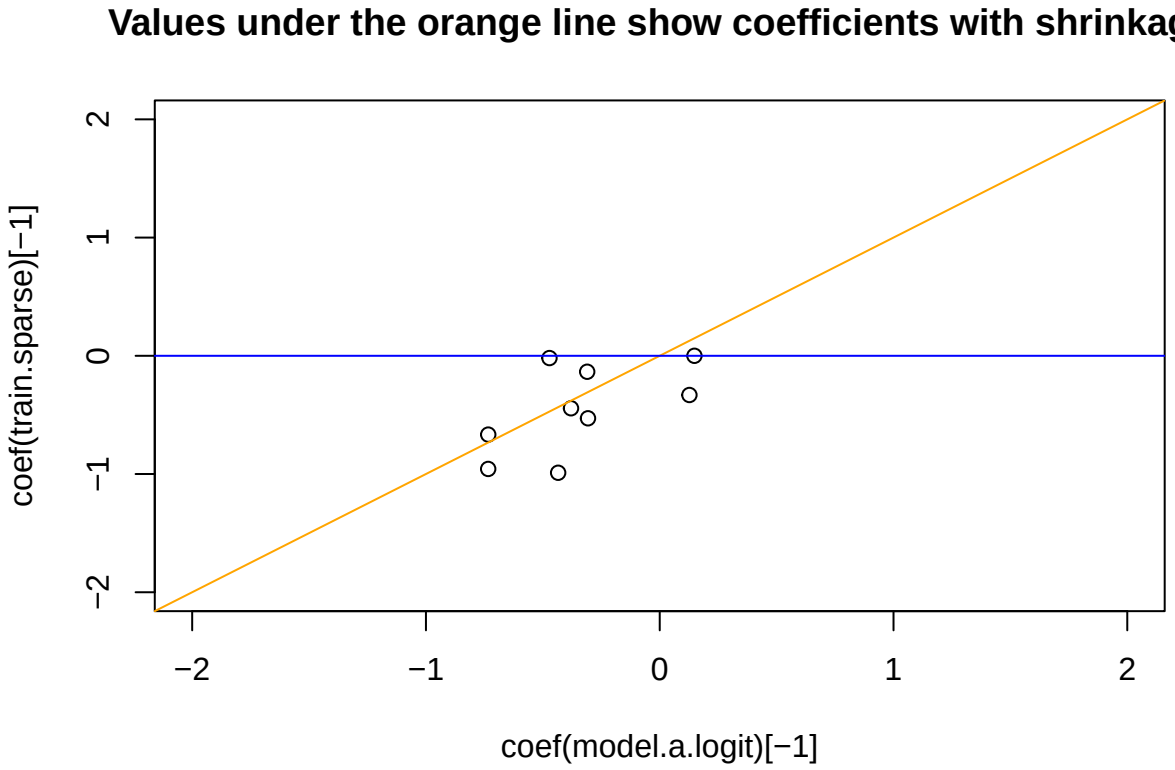
Shrinkage with maximum regularization

Per class lecture, maximum regularization occurs when we use "lambda.1se" – the value of lambda that gives the most regularized model with error within one standard error of the minimum:

```
train.sparse = glmnet(train.scaled, Class, alpha = 1, family = "binomial", lambda = train.lasso.cv$lambd
coef(train.sparse)
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept)  1.16923653
## Adhes       -0.13490727
## BNucl       -0.98923670
## Chrom       -0.66661245
## Epith        .
## Mitos       -0.01914623
## NNucl       -0.44393905
## Thick       -0.95711584
## UShap       -0.52861132
## USize       -0.33164808
```

```
# confirm shrinkage by comparison with model from part a
main = 'Values under the orange line show coefficients with shrinkage'
plot(coef(model.a.logit)[-1], coef(train.sparse)[-1], ylim=c(-2,2), xlim=c(-2,2), main=main)
abline(0, 1, col = "orange")
abline(h = 0, col = "blue")
```




```
print('Accuracy from Lasso:')
```

```
## [1] "Accuracy from Lasso:"
```

```
get_accuracy(train.sparse, as.matrix(test2.scaled))
```

```
## Accuracy
##     0.97
```

Accuracy did not imporove. This model removed Epith predictor. Contrary to assumptions, it did not produce the simpler model, or the best accuracy.

Conclusion

All models predict with 0.975 accuracy, i.e., 97.5% of the observations in the test dataset are predicted correctly. The best model was the one obtained with the lasso, since it produced the best accuracy results with 7 predictors only, as opposed to first model, which needed all 9 predictors to produce the same accuracy, or the last model, which needed 8.

Problem 3

This problem explores to what extent attitudes toward immigration explain the switching of votes of 2012 Obama supporters to Trump in the 2016 elections.

- Dataset: 2016 Cooperative Congressional Election Study
- Only voters who voted for Obama in 2012 are in scope
- Only voters who took post-election survey are in scope
- Response variable: The respondent's vote in the 2016 presidential election (CC16_326)
- Predictors: survey weights, demographics data, immigration variables
- Method: Logistic regression is the required method

(a) Load and pre-process data

```
election.in = load.Rdata('CCES16_Common_OUTPUT_Feb2018_VV.RData','prob3')
# only keep respondents who took post-election survey
election = subset(election.in, tookpost=='Yes')
# only keep respondents who voted for Obama in 2012
election = subset(election, CC16_326 == 'Barack Obama')
# save education levels as character; variable will be used for plotting
educ_levels = as.character(levels(election$educ))[1:6]
# save party preferences levels as character; variable will be used for plotting
party_pref = as.character(levels(election$pid7))[1:8]
# create bin variable = 1 if respondent voted for Trump and 0 otherwise
trump_voter = ifelse(election$CC16_410a=='Donald Trump (Republican)', 1, 0)
trump_voter[is.na(trump_voter)] = 0
election$trump_voter = trump_voter
# create quantitative variable counting number of favorable responses towards immigration
# note that for some questions 'Yes' is a favorable answer, for others it's the opposite
election$imm_favorable = recode(election$CC16_331_1, 'Yes' = 1, 'No' = 0) +
                         recode(election$CC16_331_2, 'Yes' = 0, 'No' = 1) +
                         recode(election$CC16_331_3, 'Yes' = 1, 'No' = 0) +
                         recode(election$CC16_331_7, 'Yes' = 0, 'No' = 1)
# reduce racial categories to 4
election$race = recode(election$race, 'White'='White', 'Black'='Black', 'Hispanic'='Hispanic',
                       .default='Other')
# convert education level to numeric; the highest the number the more educated
election$educ = as.numeric(election$educ)
# convert party to numeric; the highest the number the less inclined a person should be
# to vote for Trump for reasons having to do with party affilication/identification
election$pid7 = as.numeric(election$pid7)
# keep only relevant variables
relevant_vars = c('trump_voter','imm_favorable','commonweight_vv_post','gender','educ','race','pid7')
election = election[,relevant_vars]
# overview of data
str(election)
```

```
## 'data.frame':    23395 obs. of  7 variables:
##  $ trump_voter         : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ imm_favorable       : num  4 1 4 4 2 2 3 1 4 2 ...
##  $ commonweight_vv_post: num  1.093 0.543 1.112 1.078 0.363 ...
##  $ gender              : Ord.factor w/ 4 levels "Male"<"Female"<..: 2 2 2 1 2 1 1 2 1 2 ...
##  $ educ                : num  5 2 5 3 5 3 4 2 2 6 ...
```

```
##  $ race                : Ord.factor w/ 4 levels "White"<"Black"<..: 1 4 4 4 3 3 1 1 1 1 ...
##  $ pid7                : num  2 2 2 1 2 1 4 2 1 3 ...
```

```
summary(election)
```

```
##   trump_voter      imm_favorable   commonweight_vv_post       gender
##  Min.   :0.00000   Min.   :0.000   Min.   : 0.0001      Male     : 9864
##  1st Qu.:0.00000   1st Qu.:2.000   1st Qu.: 0.4294      Female   :13531
##  Median :0.00000   Median :3.000   Median : 0.6564      Skipped  :    0
##  Mean   :0.09066   Mean   :2.806   Mean   : 0.9122      Not Asked:    0
##  3rd Qu.:0.00000   3rd Qu.:4.000   3rd Qu.: 0.9879
##  Max.   :1.00000   Max.   :4.000   Max.   :15.0003
##
##       educ             race            pid7
##  Min.   :1.000   White   :16170   Min.   :1.00
##  1st Qu.:3.000   Black   : 3795   1st Qu.:1.00
##  Median :4.000   Hispanic: 1770   Median :2.00
##  Mean   :4.022   Other   : 1660   Mean   :2.16
##  3rd Qu.:5.000                    3rd Qu.:3.00
##  Max.   :6.000                    Max.   :8.00
##                                   NA's   :18
```

```
# confirm counts match Dr. Luen's
summary(as.factor(election$trump_voter))
```

```
##     0     1
## 21274  2121
```

```
ggpairs(election, progress=F)
```

```
## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 18 rows containing missing values
```

```
## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 18 rows containing missing values
```

```
## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 18 rows containing missing values
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 18 rows containing non-finite values (stat_boxplot).
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning in (function (data, mapping, alignPercent = 0.6, method =
## "pearson", : Removed 18 rows containing missing values
```

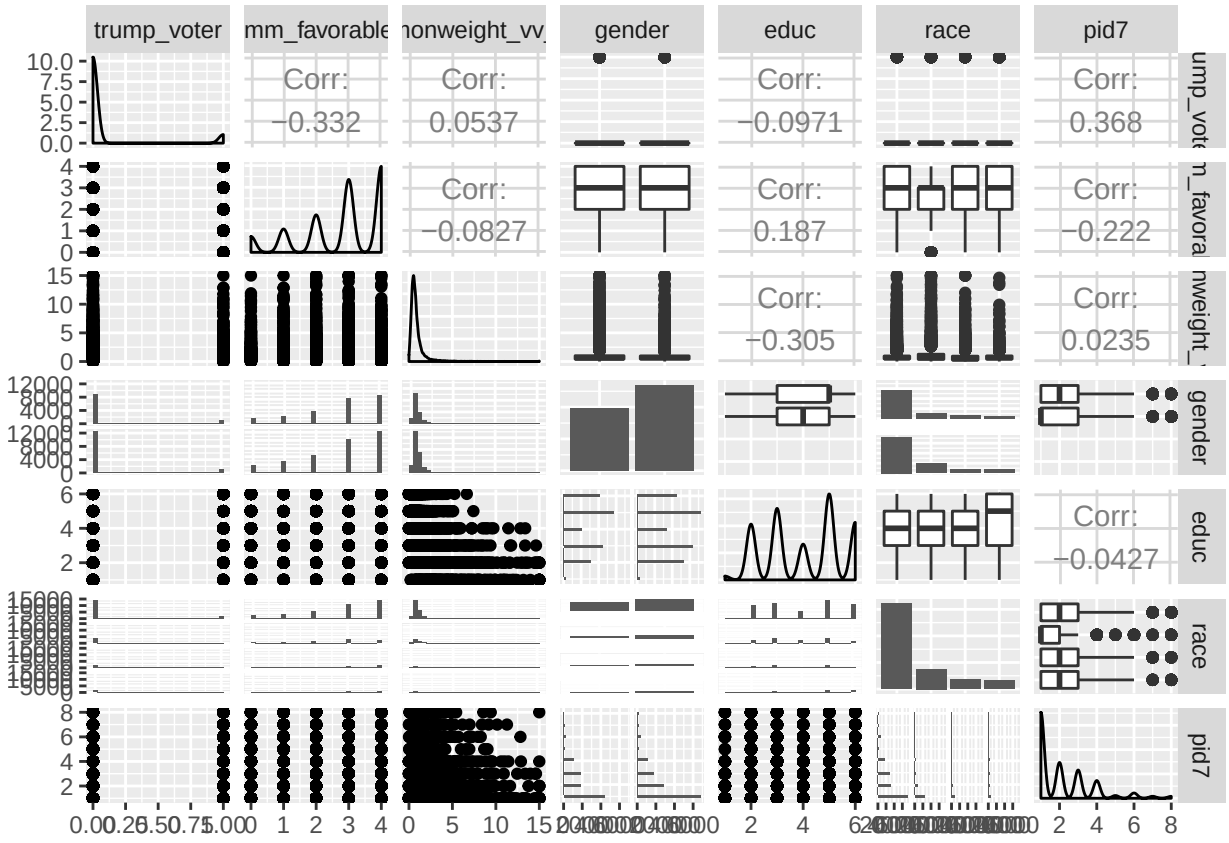
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 18 rows containing non-finite values (stat_boxplot).
```

```
## Warning: Removed 18 rows containing missing values (geom_point).
```

```
## Warning: Removed 18 rows containing missing values (geom_point).
```

```
## Warning: Removed 18 rows containing missing values (geom_point).
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## Warning: Removed 18 rows containing non-finite values (stat_bin).
## Warning: Removed 18 rows containing missing values (geom_point).
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## Warning: Removed 18 rows containing non-finite values (stat_bin).
## Warning: Removed 18 rows containing non-finite values (stat_density).
```



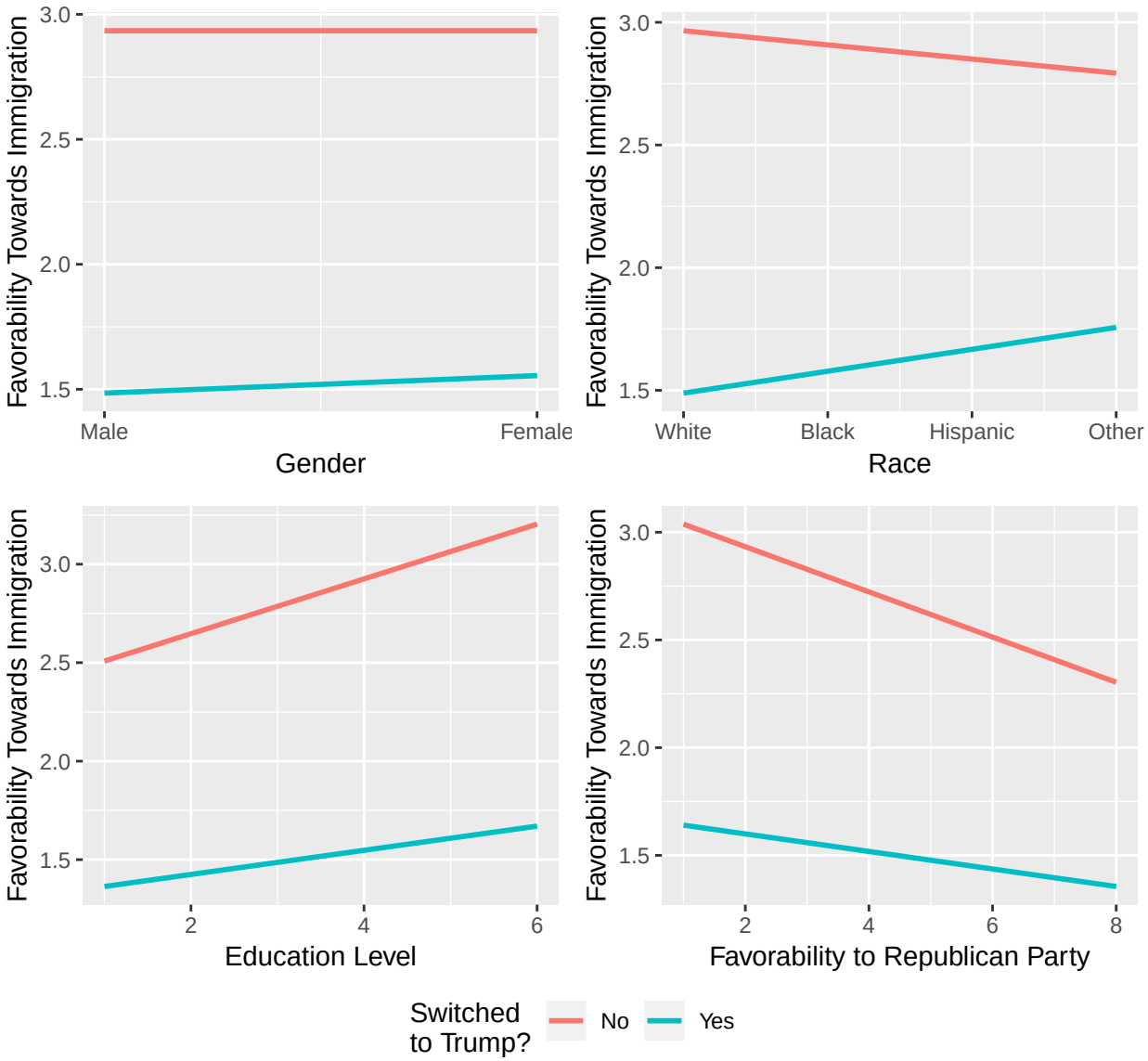
(b) Models combining immigration attitude with demographics

I will use regular logistic regression here, as opposed to weigthed logistic regression. I will not scale variables because I will not use regularized regression models.

Firt, I try to identify strong effect of interaction between demographic variables and attitude towards immigration.

EDA

```
e = na.omit(election)
# explore interaction of immigration attitude and gender
g1 = ggplot(e, aes(x = as.numeric(gender), y = imm_favorable,
                   color = factor(trump_voter, labels=c('No','Yes'))))
g1 = g1 + geom_smooth(method = "lm", se = FALSE)
g1 = g1 + ylab("Favorability Towards Immigration") + labs(col='Switched\nto Trump?') + xlab('Gender')
g1 = g1 + scale_x_continuous(breaks=c(1:2), labels=c('Male','Female'))

# explore interaction of immigration attitude and race
g2 = ggplot(e, aes(x = as.numeric(race), y = imm_favorable,
                   color = factor(trump_voter, labels=c('No','Yes'))))
g2 = g2 + geom_smooth(method = "lm", se = FALSE)
g2 = g2 + ylab("Favorability Towards Immigration") + labs(col='Switched\nto Trump?') + xlab('Race')
g2 = g2 + scale_x_continuous(breaks=c(1:4), labels=c('White','Black','Hispanic','Other'))

# explore interaction of immigration attitude and education
g3 = ggplot(e, aes(x = educ, y = imm_favorable,
                   color = factor(trump_voter, labels=c('No','Yes'))))
g3 = g3 + geom_smooth(method = "lm", se = FALSE)
g3 = g3 + ylab("Favorability Towards Immigration") + labs(col='Switched\nto Trump?') + xlab('Education l

# explore interaction of immigration attitude and party identification
g4 = ggplot(e, aes(x = pid7, y = imm_favorable,
                   color = factor(trump_voter, labels=c('No','Yes'))))
g4 = g4 + geom_smooth(method = "lm", se = FALSE)
g4 = g4 + ylab("Favorability Towards Immigration") + labs(col='Switched\nto Trump?')
g4 = g4 + xlab('Favorability to Republican Party')

ggarrange(g1, g2, g3, g4, nrow = 2, ncol = 2, legend = 'bottom', common.legend = TRUE)
```



As expected voters who did not switch to Trump in 2016 have higher favorability towards immigration on average. That doesn't change much accross genders. When it comes to other demographic data, we do notice strong interaction between attitude towards immigration and race, education level, and partidary inclination:

- Favorability towards immigration increases, on average, for non-Trump black, hispanics and other race voters; the opposite occurs to voters that switched to trump.
- Favorability towards immigration increases with education, on average, for both Trump and non-trump voters, but it increases more for Trump voters.
- Favorability towards immigration decreases the more both Trump and non-Trump voters identify with the Republican Party. The decrease is more pronounced for Trump voters.

Model 1 - Immigration and gender

For this model, I will not use interaction.

```
e = na.omit(election)
# train model using all immigration variable and gender, with no interaction
model_1 = glm(trump_voter ~ imm_favorable + gender, family = "binomial", data = e)
tidy(model_1)
```

```
## # A tibble: 3 x 5
##   term          estimate std.error statistic  p.value
##   <chr>            <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)     -0.394    0.0407     -9.69 3.47e-22
## 2 imm_favorable   -0.839    0.0186    -45.2  0.
## 3 gender.L        -0.167    0.0346     -4.84 1.30e- 6
```

```
# print accuracy
confusionMatrix(as.factor(round(model_1$fitted.values,0)), as.factor(e$trump_voter), positive='1')
```

```
## Warning in confusionMatrix.default(as.factor(round(model_1$fitted.values, :
## Levels are not in the same order for reference and data. Refactoring data
## to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 21260  2117
##          1     0     0
##
##                Accuracy : 0.9094
##                  95% CI : (0.9057, 0.9131)
##     No Information Rate : 0.9094
##     P-Value [Acc > NIR] : 0.5058
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.90944
##              Prevalence : 0.09056
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : 1
##
```

Model 2 - Immigration and race

For this model, I will use interaction.

```
e = na.omit(election)
# train model using all immigration variable and race, with interaction
model_2 = glm(trump_voter ~ imm_favorable * race, family = "binomial", data = e)
tidy(model_2)
```

```
## # A tibble: 8 x 5
##   term                 estimate std.error statistic   p.value
##   <chr>                   <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)           -0.880     0.0732   -12.0   2.90e- 33
## 2 imm_favorable         -0.755     0.0324   -23.3   3.57e-120
## 3 race.L                -0.0860    0.122     -0.708 4.79e-  1
## 4 race.Q                 0.872     0.146      5.96  2.60e-  9
## 5 race.C                -1.39      0.168     -8.26  1.42e- 16
## 6 imm_favorable:race.L   0.0222    0.0570     0.390 6.96e-  1
## 7 imm_favorable:race.Q  -0.188     0.0648    -2.90  3.72e-  3
## 8 imm_favorable:race.C   0.146     0.0717     2.03  4.22e-  2
```

```
# print accuracy
confusionMatrix(as.factor(round(model_2$fitted.values,0)), as.factor(e$trump_voter), positive='1')
```

```
## Warning in confusionMatrix.default(as.factor(round(model_2$fitted.values, :
## Levels are not in the same order for reference and data. Refactoring data
## to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 21260  2117
##          1     0     0
##
##                Accuracy : 0.9094
##                  95% CI : (0.9057, 0.9131)
##     No Information Rate : 0.9094
##     P-Value [Acc > NIR] : 0.5058
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.90944
##              Prevalence : 0.09056
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : 1
##
```

Model 3 - Immigration and education

For this model, I will use interaction.

```
e = na.omit(election)
# train model using all immigration variable and education, with interaction
model_3 = glm(trump_voter ~ imm_favorable * educ, family = "binomial", data = e)
tidy(model_3)
```

```
## # A tibble: 4 x 5
##   term                estimate std.error statistic  p.value
##   <chr>                  <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)          -0.368     0.110     -3.35  8.22e- 4
## 2 imm_favorable        -0.617     0.0516   -12.0   6.48e-33
## 3 educ                 -0.0164    0.0288    -0.569 5.69e- 1
## 4 imm_favorable:educ   -0.0548    0.0131    -4.18  2.97e- 5
```

```
# print accuracy
confusionMatrix(as.factor(round(model_3$fitted.values,0)), as.factor(e$trump_voter), positive='1')
```

```
## Warning in confusionMatrix.default(as.factor(round(model_3$fitted.values, :
## Levels are not in the same order for reference and data. Refactoring data
## to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 21260  2117
##          1     0     0
##
##                Accuracy : 0.9094
##                  95% CI : (0.9057, 0.9131)
##     No Information Rate : 0.9094
##     P-Value [Acc > NIR] : 0.5058
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.90944
##              Prevalence : 0.09056
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : 1
##
```

Model 4 - Immigration and party

For this model, I will use interaction.

```
# train model using all immigration variable and party, with interaction
e = na.omit(election)
model_4 = glm(trump_voter ~ imm_favorable * pid7, family = "binomial", data = e)
tidy(model_4)
```

```
## # A tibble: 4 x 5
##   term                estimate std.error statistic   p.value
##   <chr>                  <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)          -1.99      0.0904    -22.0  3.77e-107
## 2 imm_favorable        -0.820     0.0394    -20.8  5.34e- 96
## 3 pid7                  0.497     0.0247     20.2  2.02e- 90
## 4 imm_favorable:pid7    0.0215    0.0106      2.03 4.21e-  2
```

```
# print accuracy
confusionMatrix(as.factor(round(model_4$fitted.values,0)), as.factor(e$trump_voter), positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 20826  1515
##          1   434   602
##
##                Accuracy : 0.9166
##                  95% CI : (0.913, 0.9201)
##     No Information Rate : 0.9094
##     P-Value [Acc > NIR] : 5.642e-05
##
##                   Kappa : 0.3427
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.28436
##             Specificity : 0.97959
##          Pos Pred Value : 0.58108
##          Neg Pred Value : 0.93219
##              Prevalence : 0.09056
##          Detection Rate : 0.02575
##    Detection Prevalence : 0.04432
##       Balanced Accuracy : 0.63198
##
##        'Positive' Class : 1
##
```

Model 5 - Immigration and all demographic variables

For this model, I will use interaction between immigration and race, education and party.

```
e = na.omit(election)
# train model using all immigration variable and party, with interaction
model_5 = glm(trump_voter ~ gender + imm_favorable * (race + educ + pid7), family = "binomial", data = e
tidy(model_5)
```

```
## # A tibble: 13 x 5
##    term                 estimate std.error statistic  p.value
##    <chr>                   <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)           -2.29      0.167    -13.8   3.55e-43
##  2 gender.L              -0.172     0.0379    -4.54  5.61e- 6
##  3 imm_favorable         -0.536     0.0733    -7.31  2.59e-13
##  4 race.L                -0.265     0.133     -2.00  4.59e- 2
##  5 race.Q                 0.579     0.158      3.67  2.46e- 4
##  6 race.C                -1.12      0.180     -6.19  6.09e-10
##  7 educ                  -0.0110    0.0327    -0.337 7.36e- 1
##  8 pid7                   0.463     0.0252    18.4   2.35e-75
##  9 imm_favorable:race.L   0.0622    0.0604     1.03  3.04e- 1
## 10 imm_favorable:race.Q  -0.130     0.0687    -1.89  5.92e- 2
## 11 imm_favorable:race.C   0.113     0.0762     1.49  1.37e- 1
## 12 imm_favorable:educ    -0.0555    0.0144    -3.84  1.21e- 4
## 13 imm_favorable:pid7     0.0289    0.0108     2.67  7.51e- 3
```

```
# print accuracy
confusionMatrix(as.factor(round(model_5$fitted.values,0)), as.factor(e$trump_voter), positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 20926  1546
##          1   334   571
##
##                Accuracy : 0.9196
##                  95% CI : (0.916, 0.923)
##     No Information Rate : 0.9094
##     P-Value [Acc > NIR] : 2.106e-08
##
##                   Kappa : 0.3422
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.26972
##             Specificity : 0.98429
##          Pos Pred Value : 0.63094
##          Neg Pred Value : 0.93120
##              Prevalence : 0.09056
##          Detection Rate : 0.02443
##    Detection Prevalence : 0.03871
##       Balanced Accuracy : 0.62701
##
##        'Positive' Class : 1
```

##

Conclusion

Models 1, 2 and 3, use information on immigration and gender (no interaction), immigration and race (with interaction) and immigration and education (with interaction). These models are not capable of fitting Trump voters. Models 4 (immigration and party) and Model 5 (immigration plus all demographic variables) are capable of fitting Trump voters. Model 5 has slightly superior accuracy, but Model 4 has the best sensitivity, and is therefore the best model to study voters which switched to Trump. I did not analyze residuals because I assumed (hopefully correctly) that it was beyond the scope of this problem.

(c) Weighted logistic regression models to predict probability of Obama voter switching to Trump

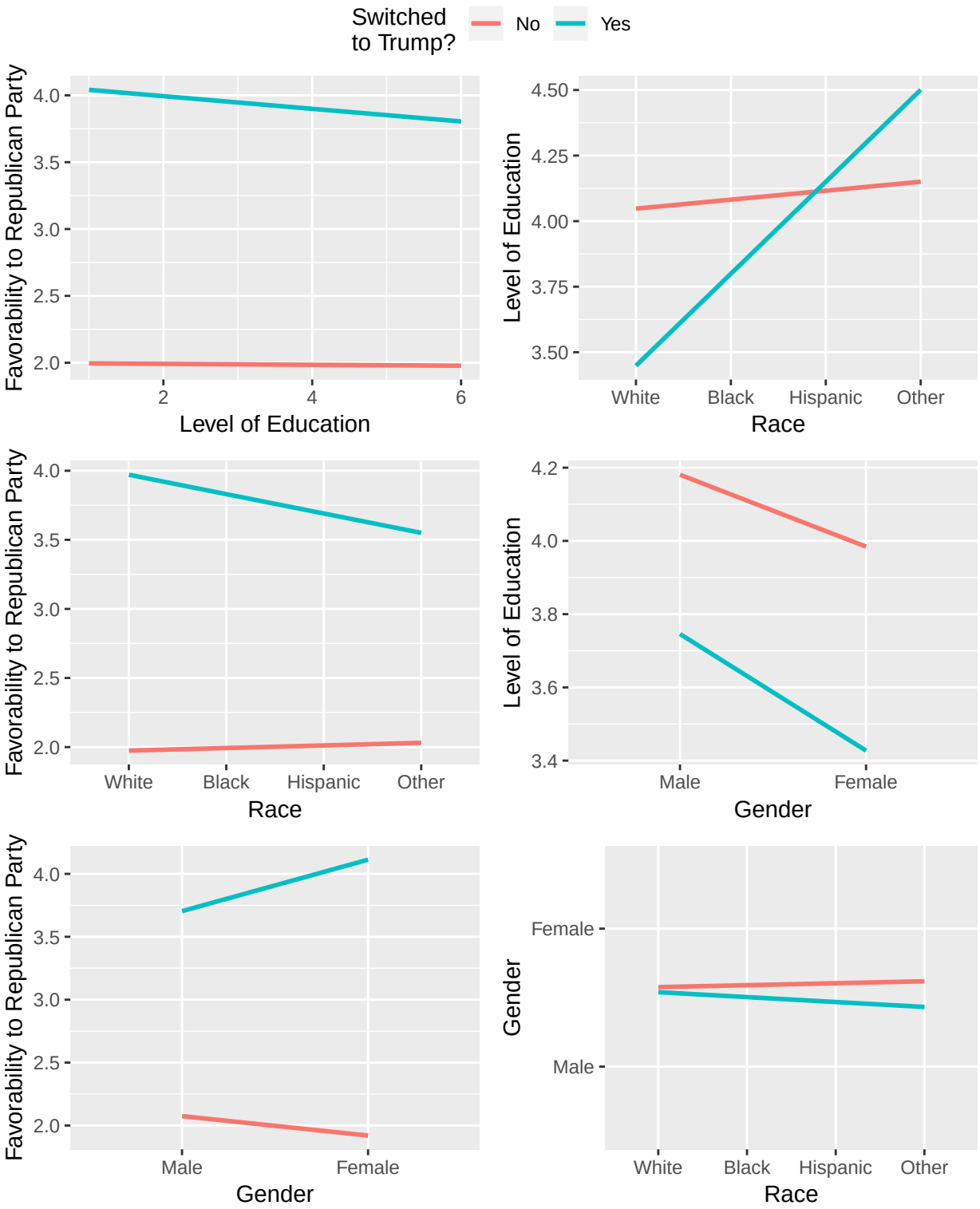
In part (b) of this problem we explored the interaction of immigration and the several demographic predictors. Before fitting a model without immigration I perform EDA to identify possible interaction between demographic variables.

EDA

Explore interactions among demographic variables.

```
e = na.omit(election)
g1 = ggplot(e, aes(x = educ, y = pid7,
                   group=trump_voter, color = factor(trump_voter, labels=c('No','Yes'))))
g1 = g1 + xlab('Level of Education') + ylab("Favorability to Republican Party")
g1 = g1 + geom_smooth(method = "lm", se = FALSE)
g1 = g1 + labs(col='Switched\nto Trump?')

g2 = ggplot(e, aes(x = race, y = educ,
                   group=trump_voter, color = factor(trump_voter, labels=c('No','Yes'))))
g2 = g2 + xlab('Race') + ylab("Level of Education")
g2 = g2 + geom_smooth(method = "lm", se = FALSE)
g2 = g2 + labs(col='Switched\nto Trump?')

g3 = ggplot(e, aes(x = gender, y = educ,
                   group=trump_voter, color = factor(trump_voter, labels=c('No','Yes'))))
g3 = g3 + xlab('Gender') + ylab("Level of Education")
g3 = g3 + geom_smooth(method = "lm", se = FALSE)
g3 = g3 + labs(col='Switched\nto Trump?')

g4 = ggplot(e, aes(x = gender, y = pid7,
                   group=trump_voter, color = factor(trump_voter, labels=c('No','Yes'))))
g4 = g4 + xlab('Gender') + ylab("Favorability to Republican Party")
g4 = g4 + geom_smooth(method = "lm", se = FALSE)
g4 = g4 + labs(col='Switched\nto Trump?')

g5 = ggplot(e, aes(x = race, y = gender,
                   group=trump_voter, color = factor(trump_voter, labels=c('No','Yes'))))
g5 = g5 + xlab('Race') + ylab("Gender")
g5 = g5 + geom_smooth(method = "lm", se = FALSE)
g5 = g5 + labs(col='Switched\nto Trump?')
```

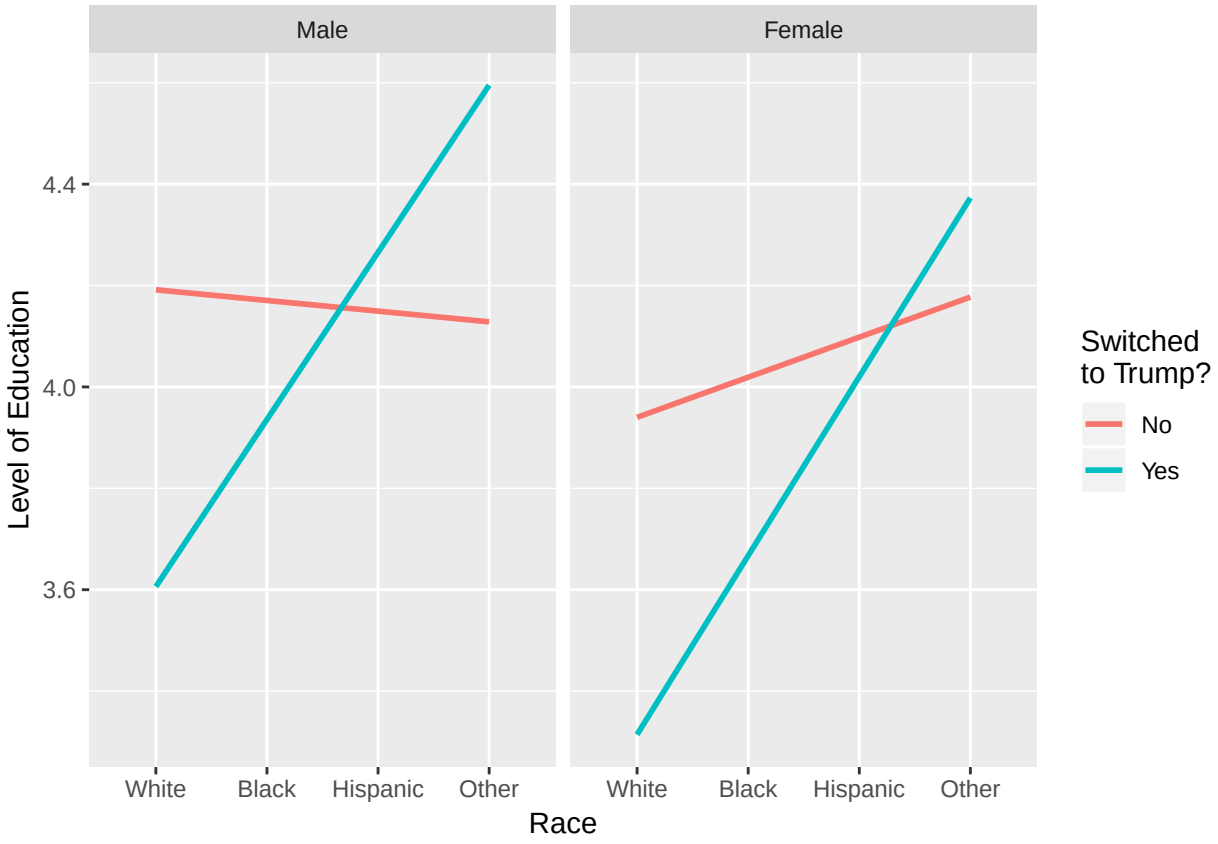
```
g6 = ggplot(e, aes(x = race, y = pid7,
                   group=trump_voter, color = factor(trump_voter, labels=c('No','Yes'))))
g6 = g6 + xlab('Race') + ylab("Favorability to Republican Party")
g6 = g6 + geom_smooth(method = "lm", se = FALSE)
g6 = g6 + labs(col='Switched\nto Trump?')

ggarrange(g1, g2, g6, g3, g4, g5, nrow = 3, ncol = 2, legend = 'top', common.legend = TRUE)
```



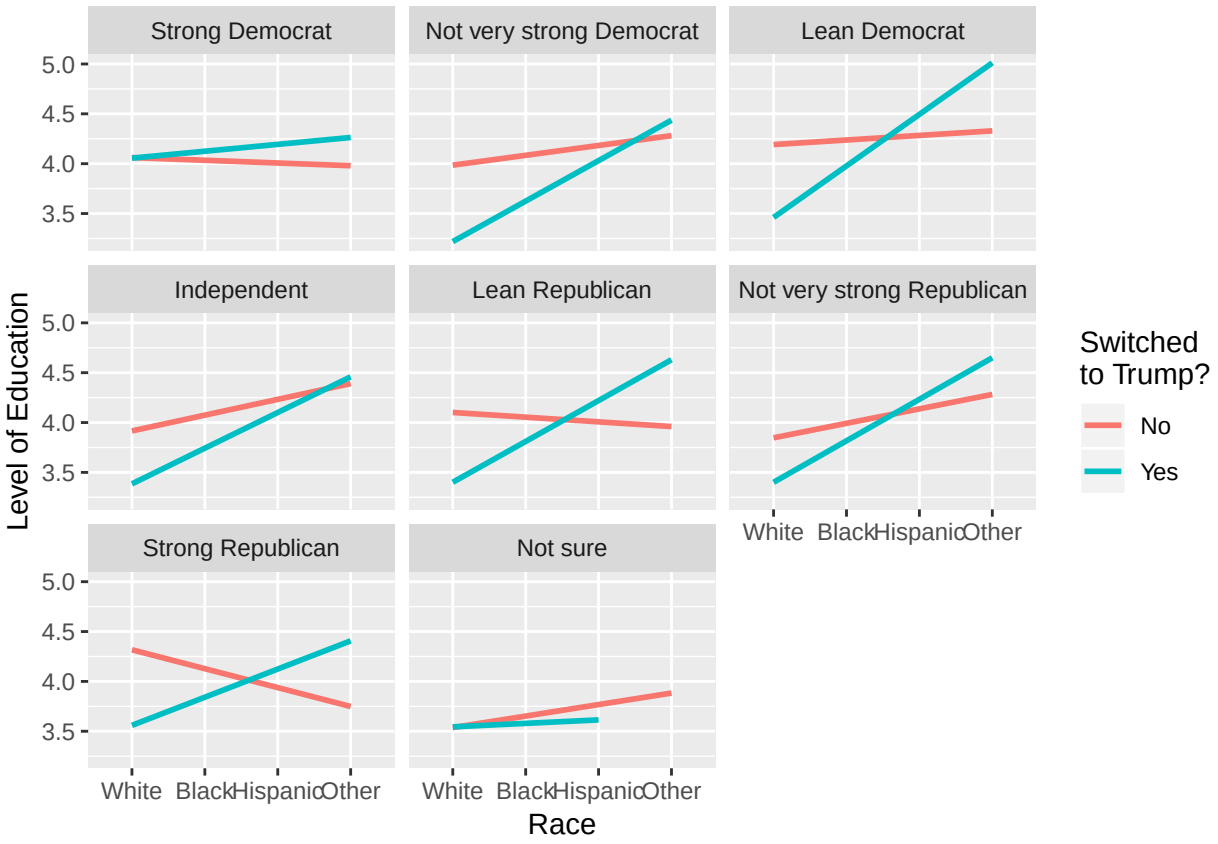
We notice a strong interaction between level of education and race. Level of eduation did not matter much for non-Trump voters, but blacks, hispanic and other races tended to vote for Trump in 2016 the more educated they were.

I will now see if these 2 predictors further interact with gender and party affiliation.

```
g1 = ggplot(e, aes(x = race, y = educ,
                   group=trump_voter, color = factor(trump_voter, labels=c('No','Yes'))))
g1 = g1 + geom_smooth(method = "lm", se = FALSE)
g1 = g1 + xlab('Race') + ylab("Level of Education")
g1 = g1 + labs(col='Switched\nto Trump?')
g1 = g1 + facet_wrap(e$gender)
g1
```




```
g2 = ggplot(e, aes(x = race, y = educ,
                   group=trump_voter, color = factor(trump_voter, labels=c('No','Yes'))))
g2 = g2 + geom_smooth(method = "lm", se = FALSE)
g2 = g2 + xlab('Race') + ylab("Level of Education")
g2 = g2 + labs(col='Switched\nto Trump?')
g2 = g2 + facet_wrap(factor(e$pid7, labels = party_pref))
g2
```



In the first plot we notice that the interaction between level of education and race doesn't change that much with gender. In the second plot we notice a change in the relationship for strong democrats, and "not sure" groups. For these groups the race didn't seem interact much with their education across Trump and non-Trump voter groups. For that reason, I will include the interaction of level of education, race AND party in my first weighted model. In my second model, I will make all variables interact to make the most of all information available in the data.

Model c.1 Weighted logistic regression without immigration attitude as predictor

```
e = na.omit(election)
model.c.1 = glm(trump_voter ~ gender + (educ * race * pid7), family = quasibinomial,
                weights = commonweight_vv_post, data = e)
tidy(model.c.1)
```

```
## # A tibble: 17 x 5
##    term             estimate std.error statistic  p.value
##    <chr>               <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)       -3.56      0.200    -17.8   1.39e-70
##  2 gender.L          -0.196     0.0327    -5.98  2.21e- 9
##  3 educ              -0.114     0.0522    -2.19  2.86e- 2
##  4 race.L            -1.24      0.457     -2.72  6.56e- 3
##  5 race.Q            -0.568     0.399     -1.42  1.55e- 1
##  6 race.C            -1.14      0.332     -3.44  5.75e- 4
##  7 pid7               0.390     0.0468     8.35  7.46e-17
##  8 educ:race.L        0.272     0.105      2.58  9.94e- 3
##  9 educ:race.Q        0.144     0.104      1.37  1.70e- 1
## 10 educ:race.C        0.201     0.104      1.94  5.26e- 2
## 11 educ:pid7          0.0394    0.0130     3.03  2.47e- 3
## 12 race.L:pid7       -0.0833    0.0970    -0.859 3.90e- 1
## 13 race.Q:pid7        0.396     0.0935     4.23  2.36e- 5
## 14 race.C:pid7        0.100     0.0899     1.11  2.65e- 1
## 15 educ:race.L:pid7   0.0150    0.0239     0.629 5.29e- 1
## 16 educ:race.Q:pid7  -0.0767    0.0260    -2.95  3.23e- 3
## 17 educ:race.C:pid7  -0.0524    0.0280    -1.87  6.16e- 2
```

```
confusionMatrix(as.factor(round(model.c.1$fitted.values,0)), as.factor(e$trump_voter), positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 20920  1713
##          1   340   404
##
##                Accuracy : 0.9122
##                  95% CI : (0.9085, 0.9158)
##     No Information Rate : 0.9094
##     P-Value [Acc > NIR] : 0.07344
##
##                   Kappa : 0.247
##
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.19084
##             Specificity : 0.98401
##          Pos Pred Value : 0.54301
##          Neg Pred Value : 0.92431
##              Prevalence : 0.09056
##          Detection Rate : 0.01728
##    Detection Prevalence : 0.03183
##       Balanced Accuracy : 0.58742
```

```
##
##        'Positive' Class : 1
##
```

Below I display model mean probabilities of switching to Trump for different genders and races.

```
e$probs = model.c.1$fitted.values
e %>% select(gender, probs) %>% group_by(gender) %>% summarize(mean(probs))
```

```
## # A tibble: 2 x 2
##   gender `mean(probs)`
##   <ord>          <dbl>
## 1 Male          0.113
## 2 Female        0.0908
```

```
e %>% select(race, probs) %>% group_by(race) %>% summarize(mean(probs))
```

```
## # A tibble: 4 x 2
##   race     `mean(probs)`
##   <ord>            <dbl>
## 1 White           0.118
## 2 Black           0.0354
## 3 Hispanic        0.0949
## 4 Other           0.0803
```

Model c.2 Weighted logistic regression with immigration attitude as predictor

```
e = na.omit(election)
# train model using all immigration variable and party, with interaction
model.c.2 = glm(trump_voter ~ gender + imm_favorable * (race * educ * pid7), family = quasibinomial,
                weights = commonweight_vv_post, data = e)
tidy(model.c.2)
```

```
## # A tibble: 33 x 5
##    term          estimate std.error statistic  p.value
##    <chr>            <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)   -2.68       0.353   -7.60    3.13e-14
##  2 gender.L      -0.192      0.0346  -5.56    2.71e- 8
##  3 imm_favorable -0.427      0.161   -2.65    8.07e- 3
##  4 race.L        -0.520      0.767   -0.678   4.98e- 1
##  5 race.Q         0.00142    0.707    0.00200 9.98e- 1
##  6 race.C        -0.864      0.641   -1.35    1.78e- 1
##  7 educ           0.136      0.0959   1.41    1.58e- 1
##  8 pid7           0.473      0.0876   5.39    7.03e- 8
##  9 race.L:educ    0.139      0.178    0.781   4.35e- 1
## 10 race.Q:educ    0.112      0.192    0.582   5.61e- 1
## # ... with 23 more rows
```

```
# print accuracy
confusionMatrix(as.factor(round(model.c.2$fitted.values,0)), as.factor(e$trump_voter), positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 20867  1468
##          1   393   649
##
##                Accuracy : 0.9204
##                  95% CI : (0.9168, 0.9238)
##     No Information Rate : 0.9094
##     P-Value [Acc > NIR] : 1.505e-09
##
##                   Kappa : 0.3735
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.30657
##             Specificity : 0.98151
##          Pos Pred Value : 0.62284
##          Neg Pred Value : 0.93427
##              Prevalence : 0.09056
##          Detection Rate : 0.02776
##    Detection Prevalence : 0.04457
##       Balanced Accuracy : 0.64404
##
##        'Positive' Class : 1
##
```

Below I display model mean probabilities of switching to Trump for different genders and races.

```r
e$probs = model.c.2$fitted.values
e %>% select(gender, probs) %>% group_by(gender) %>% summarize(mean(probs))
```

```
## # A tibble: 2 x 2
##   gender `mean(probs)`
##   <ord>          <dbl>
## 1 Male          0.112
## 2 Female        0.0894
```

```r
e %>% select(race, probs) %>% group_by(race) %>% summarize(mean(probs))
```

```
## # A tibble: 4 x 2
##   race     `mean(probs)`
##   <ord>            <dbl>
## 1 White           0.117
## 2 Black           0.0357
## 3 Hispanic        0.0913
## 4 Other           0.0769
```

Model Comparison

Model c.2, with 32 coefficients has better accuracy, 92.04% vs. 91.22%. The anova comparison between models (below) shows that there is evidence to conclude that the models are different, i.e., including immigration does improve the model. Model c.2 also has the best sensitivity of all models analyzed in Problem 3, including part (b).

What the sensitivity (30.66%) of the model tells us is that it is only capable of correctly identifying 3 out of 10 voters who stiched to trump.

```
comparison = anova(model.c.1, model.c.2, test="Chisq")
comparison
```

```
## Analysis of Deviance Table
##
## Model 1: trump_voter ~ gender + (educ * race * pid7)
## Model 2: trump_voter ~ gender + imm_favorable * (race * educ * pid7)
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1     23360      11609
## 2     23344      10240 16   1369.4 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Does this difference in model performance matter more for certain demographic groups?

To answer this question I will run the best model (c.2) against subsets for the dataset and compare sensitivity numbers. The demographics I'm interested in are: gender and race.

Below I compare the two models for the different genders and races, one at a time.

```
get_sensitivity_improv = function(model1, model2, df) {
  without_imm = confusionMatrix(as.factor(round(model1$fitted.values,0)),
                                as.factor(df$trump_voter), positive='1')$byClass[1]
  with_imm = confusionMatrix(as.factor(round(model2$fitted.values,0)),
                             as.factor(df$trump_voter), positive='1')$byClass[1]
  return(round(with_imm/without_imm - 1,4)*100)
}

get_model1_gender = function(df) {
  return(glm(trump_voter ~ (educ * race * pid7), family = quasibinomial,
             weights = commonweight_vv_post, data = df))
}

get_model2_gender = function(df) {
  return(glm(trump_voter ~ imm_favorable * (race * educ * pid7), family = quasibinomial,
             weights = commonweight_vv_post, data = df))
}

get_model1_race = function(df) {
  return(glm(trump_voter ~ gender + (educ * pid7), family = quasibinomial,
             weights = commonweight_vv_post, data = df))
}

get_model2_race = function(df) {
  return(glm(trump_voter ~ gender + imm_favorable * (educ * pid7), family = quasibinomial,
             weights = commonweight_vv_post, data = df))
}
```

```
# analyze gender
e.male = subset(e, gender=='Male')
model1 = get_model1_gender(e.male)
model2 = get_model2_gender(e.male)
paste('Sensitivity percent improvement for Male demographic group:',
      get_sensitivity_improv(model1, model2, e.male), '%')
```

```
## [1] "Sensitivity percent improvement for Male demographic group: 99.32 %"
```

```
e.female = subset(e, gender=='Female')
model1 = get_model1_gender(e.female)
model2 = get_model2_gender(e.female)
paste('Sensitivity percent improvement for Female demographic group:',
      get_sensitivity_improv(model1, model2, e.female), '%')
```

```
## [1] "Sensitivity percent improvement for Female demographic group: 50.84 %"
```

```
# analyze race
get_improv_race_message = function(df_in, race_in) {
  df = subset(df_in, race==race_in)
  model1 = get_model1_race(df)
  model2 = get_model2_race(df)
  msg = paste('Sensitivity percent improvement for',race_in,'demographic group')
  paste(msg, get_sensitivity_improv(model1, model2, df), '%')
}

get_improv_race_message(e, 'White')
```

```
## [1] "Sensitivity percent improvement for White demographic group 55.67 %"
```

```
get_improv_race_message(e, 'Black')
```

```
## [1] "Sensitivity percent improvement for Black demographic group Inf %"
```

```
get_improv_race_message(e, 'Hispanic')
```

```
## [1] "Sensitivity percent improvement for Hispanic demographic group 42.86 %"
```

```
get_improv_race_message(e, 'Other')
```

```
## [1] "Sensitivity percent improvement for Other demographic group 1300 %"
```

Clearly, the improvement from adding immigration varies by demographic group. For example, model c.1 is not able to properly identify a single black voter who switched to Trump in 2016. Model c.2 is almost 100% superior for male voters vs. female voters.

Conclusion

Our best model includes attituted towards immigration and interactions among all variables except gender. It is a fairly complex model, with 32 coefficients. The improvement from adding immigration is significant and not the same for all demographic groups. Accuracy of the best model is a little over 92% and sensitivity about 30%. I did not analyze residuals because I assumed (hopefully correctly) that it was beyond the scope of this problem.