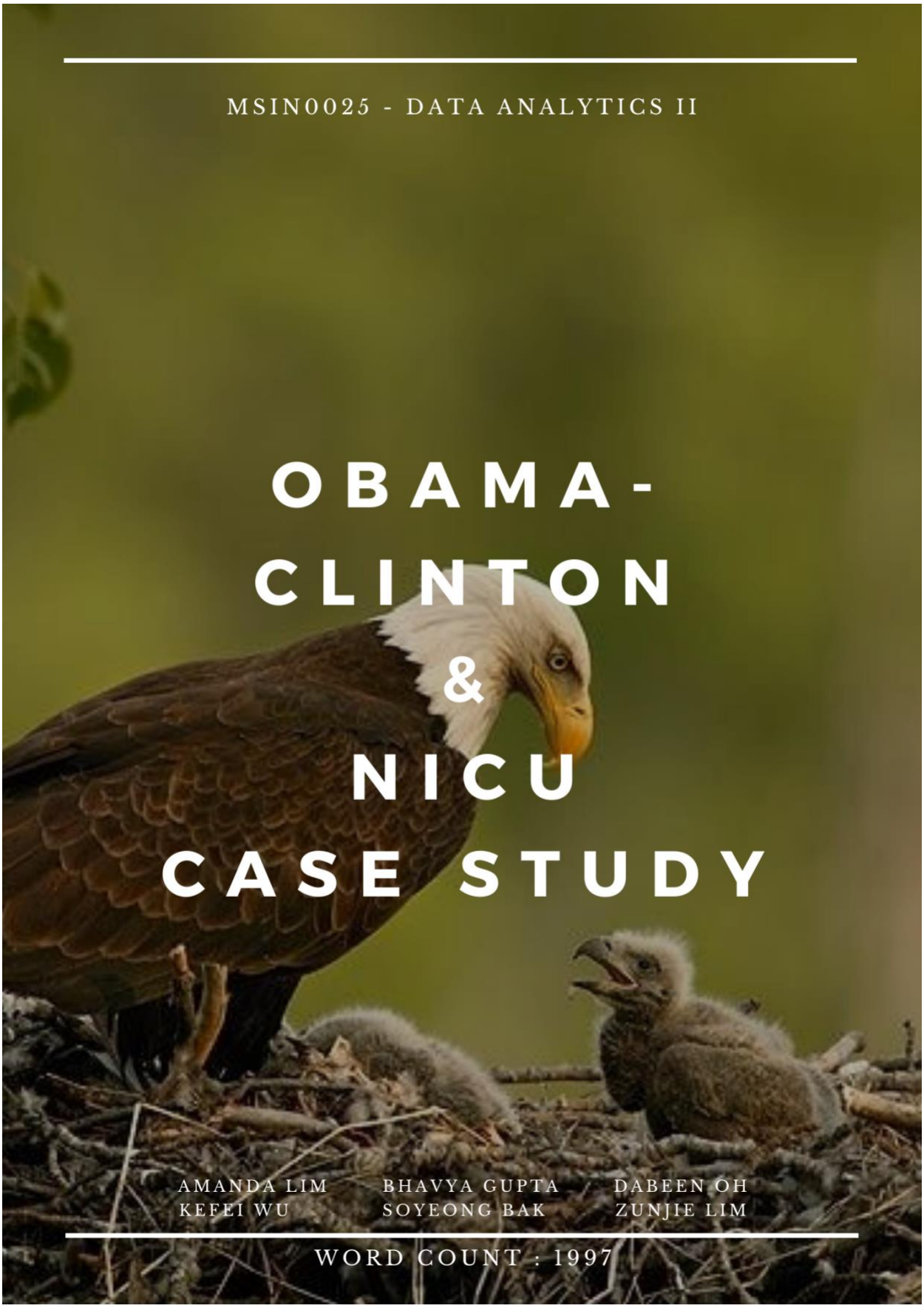
MSIN0025 - DATA ANALYTICS II

OBAMA-CLINTON & NICU CASE STUDY



AMANDA LIM
KEFEI WU
BHAVYA GUPTA
SOYEONG BAK
DABEEN OH
ZUNJIE LIM

WORD COUNT : 1997

Obama vs. Clinton Case Study

1) The Problem

In February of 2008, Barack Obama and Hilary Clinton faced off during US Democratic Party's presidential primaries – the winner of which would face the Republican party's nominee to become the next President of the United States (POTUS).

In this election process, both delegates took careless missteps in their advertising campaigns such as using misleading language or pushing marketing campaigns to audience segments that did not exist. Inaccurate audience segmentation impacts voting outcome, especially in counties where the vote is closer to 50/50.

This report will focus on what advertising strategy Obama should have taken for the remaining states by predicting the voting outcome of non-voted FIPS codes.

2) Understanding the Data

Nature of the Data

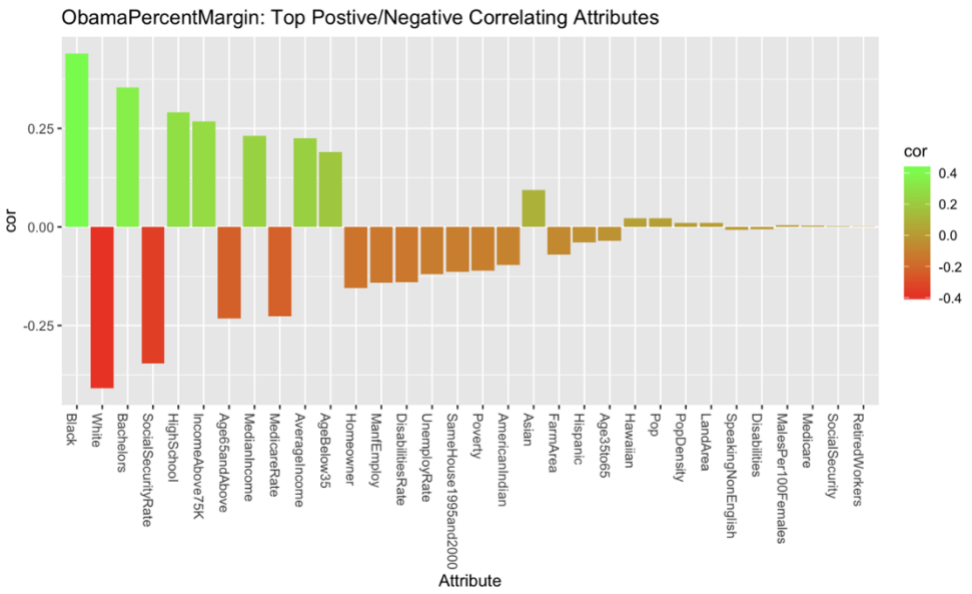
The data used contains demographic data for 2,866 unique FIPS codes in the US, 1130 of which also have voting outcome data and the remaining who do not. This dataset can be found online and thus is considered open source.

Relevant Target Attribute: Obama Percent Margin (OPM)

Obama Percent Margin is a derived attribute calculated by taking *(Votes for Obama) - (Votes for Clinton) / Total Votes*. A greater positive OPM indicates a larger lead, and a greater negative OPM indicates being further behind.

Relevance of Attributes

We created a barplot that shows the correlation between OPM and all attributes. The larger the bar, the greater the correlation.

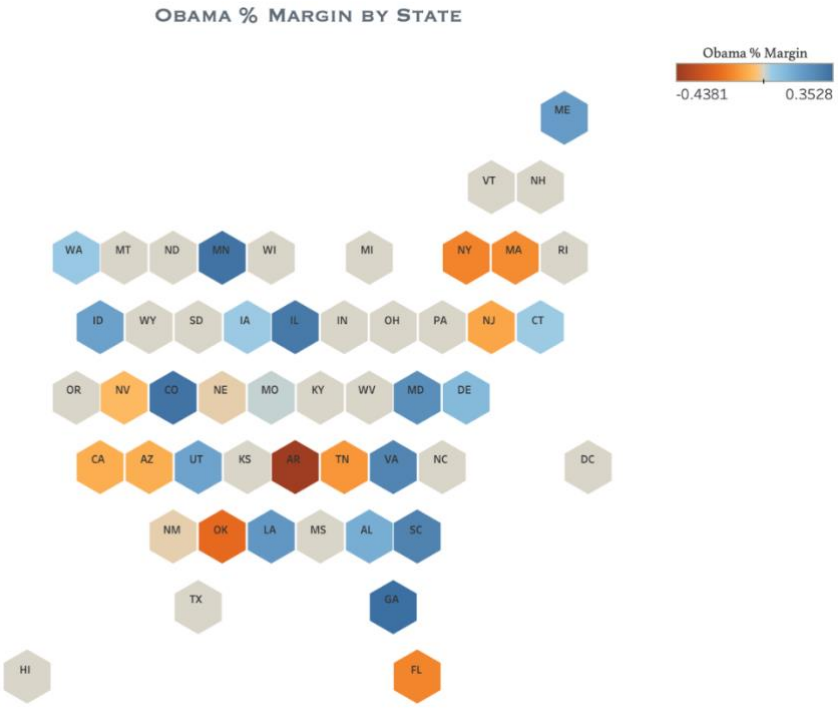


We chose the 11 attributes with the highest correlation for our analysis:

- **Race**: Black, White
- **Education**: Bachelors, HighSchool
- **Income**: IncomeAbove75k, AverageIncome, MedianIncome
- **Age**: Age65andAbove, AgeBelow35
- **Social Welfare**: SocialSecurityRate, MedicareRate

OPM by State

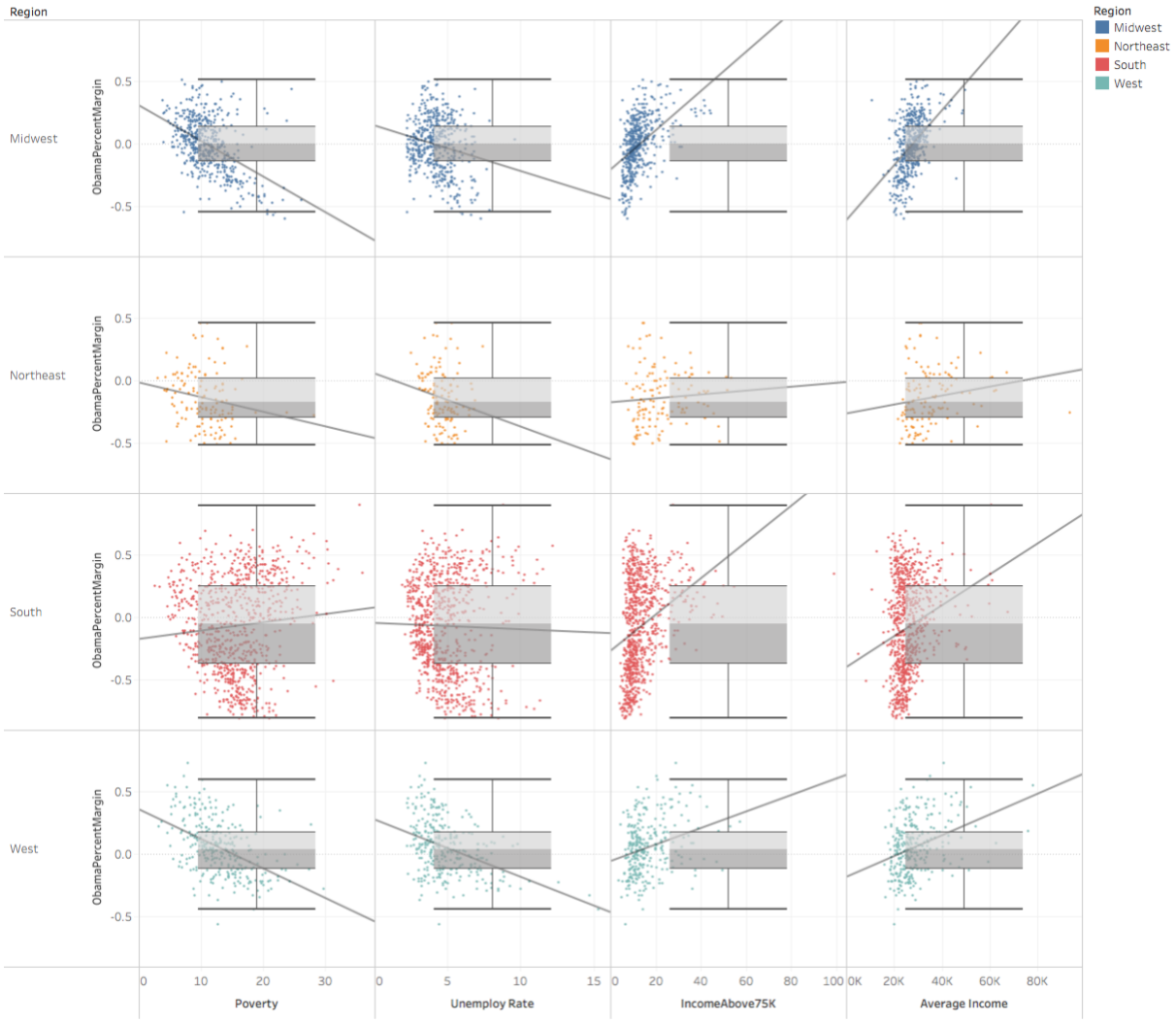
To understand the breakdown of existing OPM's in the states that have voted, we created a hex-tile map in Tableau. We chose a hex-tile over a geographical map as it eliminates misrepresentation of the significance of OPM in US states of different geographical sizes.



Our map shows that Arkansas (AR) has the most negative OPM, and Colorado (CO), Minnesota (MN), Illinois (IL), and Georgia (GA) are the most positively correlated.

OPM by Region and Income Indicators

The below visualizations show trends and distributions of poverty, unemployment rate, % of the population whose income is above 75k USD annually, and the average income.



Sum of Poverty, sum of Unemploy Rate, sum of IncomeAbove75K and sum of Average Income vs. sum of ObamaPercentMargin broken down by Region. Color shows details about Region. Details are shown for State, County and Region.

While the distribution of values does not vary much by region, the correlation with OPM does. For instance, while all regions are mostly between 20-40k for Average Income, all trend lines are positive but by different amounts, with Northeast with the flattest slope and Midwest with the steepest slope.

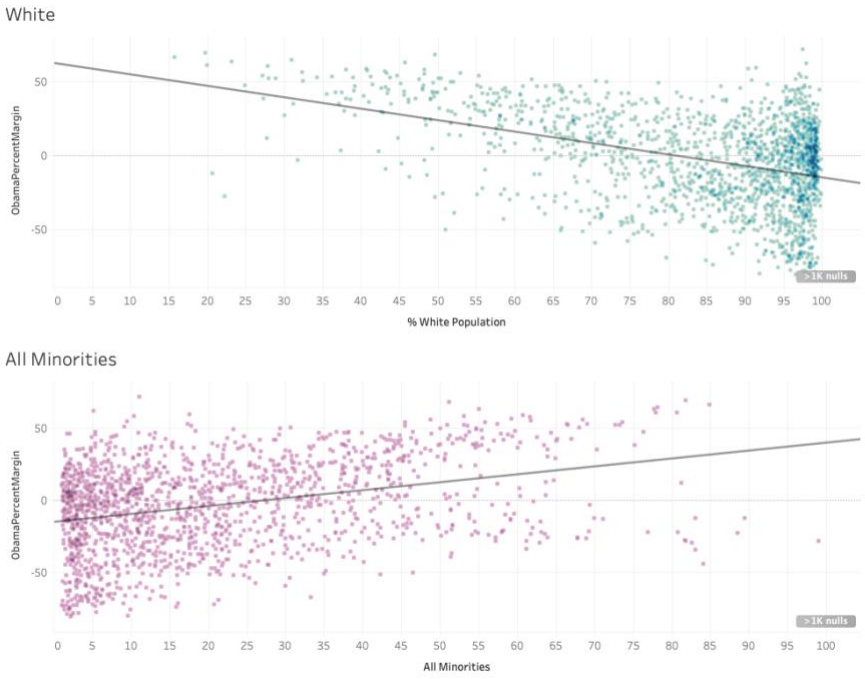
OPM by Race

We plotted the distribution of OPMs for a given-voted FIPS code based on the percentage of its population that is a given race.

White, AmericanIndian, and Hispanic are negatively correlated with OPM, and Black, Asian, and Hawaiian are positively correlated. OPM also varies greatly even within one race. For example, FIPS codes with the same percent white population have OPMs in both the extreme negative and positive ends.

Votes for Hawaii have not been counted yet, thus why % Hawaiian population only shows values less than 1.4%. Therefore, we should not consider the OPM as shown to represent the majority of the Hawaiian population.

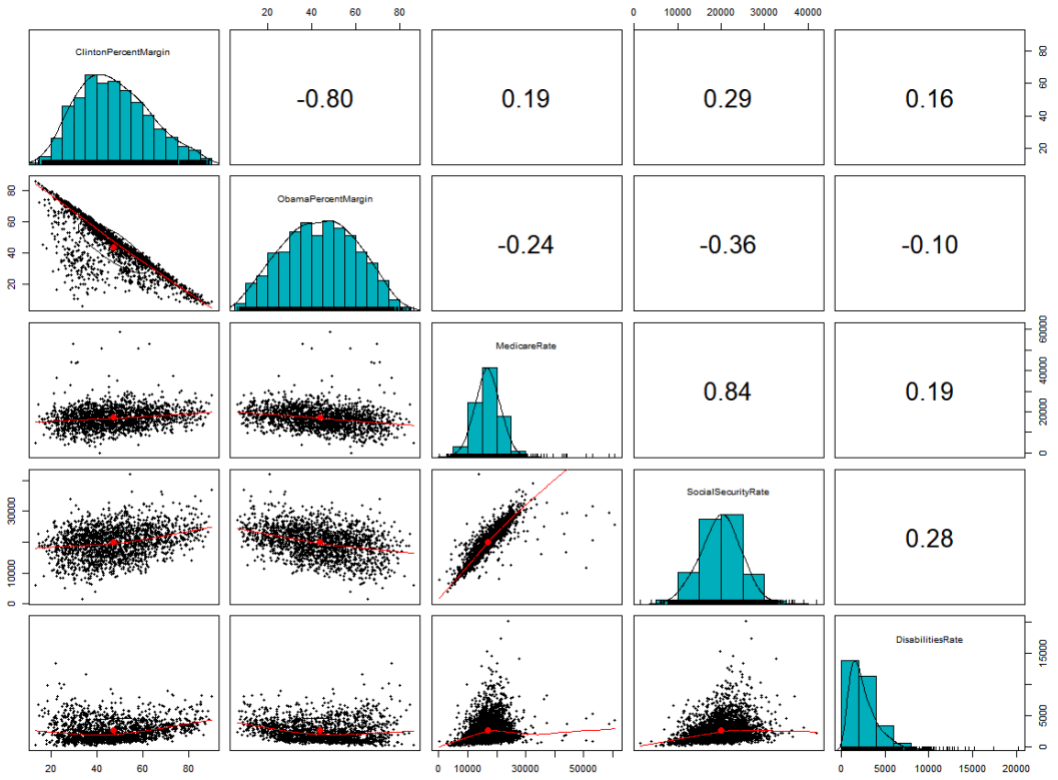
We further investigated the race attributes by grouping all minorities into a single derived attribute by adding up the percentages of all minority races in each FIPS code *(Black + American Indian + Asian + Hawaiian + Hispanic).*



The minorities attribute is positively correlated with OPM. As the percent of minorities increase, several data points deviate from the trend line. One explanation could be that FIPS codes primarily made up of one minority can give a misleading image of the OPM for all minorities.

OPM by Social Welfare

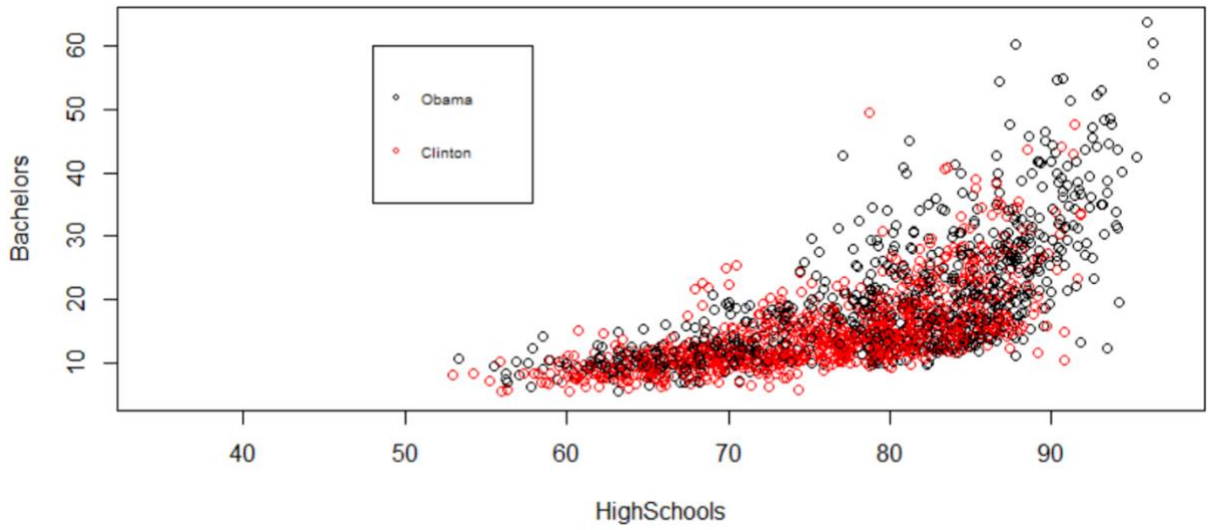
This pairwise plot shows the correlation between ClintonRate and ObamaRate by MedicareRate, SocialSecurityRate, and DisabilitiesRate.



Clinton tends to have a positive correlation with these attributes, whereas Obama tends to have a negative correlation. This means Obama appeals to the people who have low medical rates and low social security rates, however not to people with a higher percentage of people with disabilities.

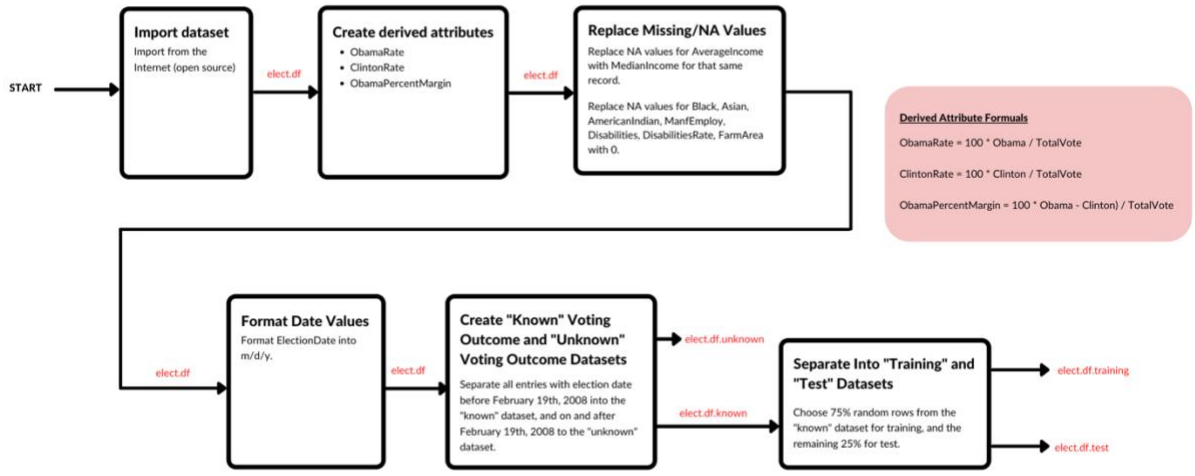
Education by Margin Leader

The below scatterplot shows FIPS codes that shows the leading candidate based on education variables. As we can see, Obama is leading in areas with higher percentages of high school and bachelor graduates.



3) Prepare the Data

The below diagram shows our process of preparing the data and the formulas for our derived attributes.



We also identified the number of missing entries from each attribute.

```
          TotalVote              Clinton               Obama
               1131                 1131                1131
              Black                Asian       AmericanIndian
                 80                   94                  99
         HighSchool            Bachelors             Poverty
                  1                    1                   1
      IncomeAbove75K       MedianIncome       AverageIncome
                  2                    1                  30
       UnemployRate           ManfEmploy   SpeakingNonEnglish
                  1                  293                   1
           Medicare        MedicareRate       SocialSecurity
                  1                    1                   1
  SocialSecurityRate      RetiredWorkers         Disabilities
                  1                    1                   8
   DisabilitiesRate            Homeowner SameHouse1995and2000
                  8                    2                   1
           LandArea             FarmArea           ObamaRate
                  1                   87                1131
         ClintonRate  ObamaPercentMargin
                1131                1131
```

4) Prediction Models

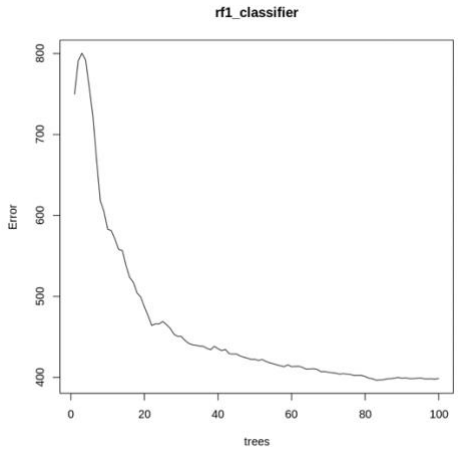
Linear Regression

We choose linear regression as it can model multiple variables, regardless of whether it is continuous or categorical. It provides coefficients for each variable and their significance, which helps decision-makers identify the most important variables. There are several ways to improve linear regression model, such as detecting the multicollinearity by identifying its Variance Inflation Factor (VIF). We also can fit an isotonic regression to remove any assumptions of the target variables form.

```
Call:
lm(formula = ObamaPercentMargin ~ AverageIncome + MedianIncome +
    Black + White + HighSchool + Bachelors + MedicareRate + SocialSecurityRate +
    IncomeAbove75K + AgeBelow35 + Age65andAbove, data = elect.df.training)

Residuals:
    Min      1Q  Median      3Q     Max
-67.856 -12.434   0.392  13.795  69.058

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)       -1.429e+02  2.098e+01  -6.812 1.47e-11 ***
AverageIncome     -3.577e-04  1.534e-04  -2.331  0.01991 *
MedianIncome       7.997e-04  2.084e-04   3.838  0.00013 ***
Black              1.709e+00  1.184e-01  14.437  < 2e-16 ***
White              3.302e-01  1.201e-01   2.750  0.00604 **
HighSchool         1.624e+00  1.267e-01  12.815  < 2e-16 ***
Bachelors          6.950e-01  1.584e-01   4.387 1.24e-05 ***
MedicareRate       3.672e-04  2.225e-04   1.650  0.09912 .
SocialSecurityRate -2.380e-03  3.645e-04  -6.532 9.33e-11 ***
IncomeAbove75K     -1.447e+00  2.952e-01  -4.903 1.07e-06 ***
AgeBelow35         -4.176e-01  2.028e-01  -2.060  0.03964 *
Age65andAbove       9.434e-01  3.996e-01   2.361  0.01837 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

        Residual standard error: 20.6 on 1290 degrees of freedom
        Multiple R-squared:  0.5275,    Adjusted R-squared:  0.5235
        F-statistic: 130.9 on 11 and 1290 DF,  p-value: < 2.2e-16
```

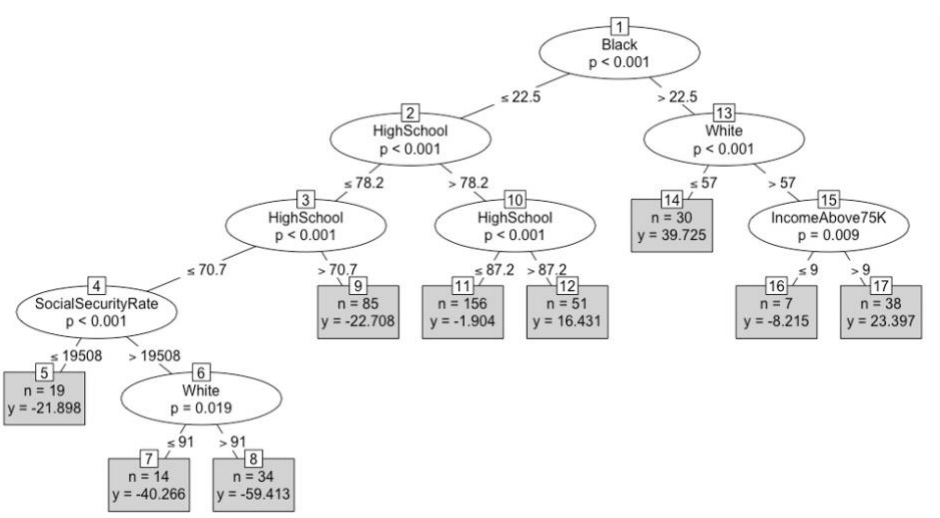
Random Forest

We chose a random forest model as it generates a more accurate result than regression trees by introducing a random component in the tree building process, which lowers the variance of a single tree's prediction. Random Forest also requires very little hyper-parameter tuning.



The plotted error rate above is based on the OOB sample error and shows that 83 trees provide the lowest error rate and an average Obama percent margin error of 19.91. The error rate can be slightly improved by tuning our random forest model.

The conditional Inference tree below shows the most important features which are Black, White, High School, Social Security Rate and Income Above 75K.

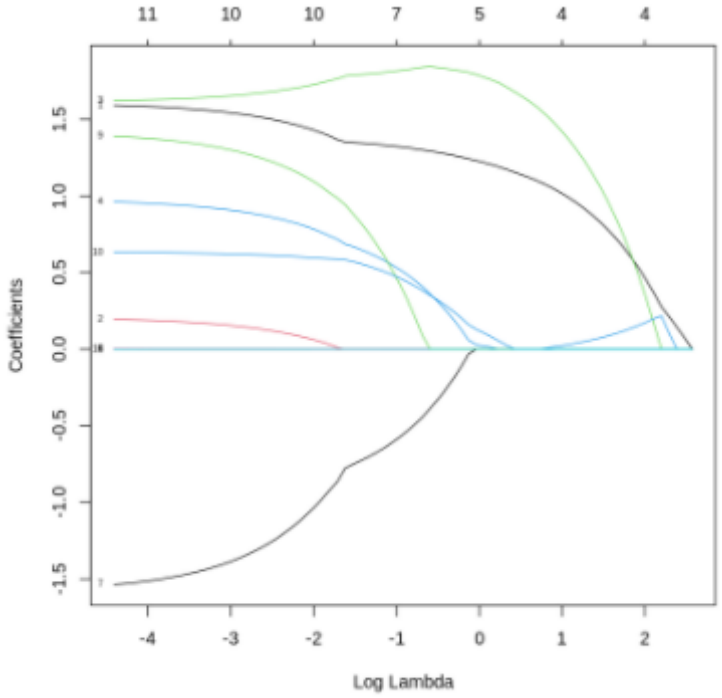


Our random forest was tuned using the grid search across the following hyperparameters: the number of trees and variables randomly sampled at each split, the sample training size, the minimum number of samples within the terminal nodes and the maximum number of terminal nodes.

LASSO Regularization

We chose a LASSO model in order to improve accuracy due to feature selection from a large set of attributes. Each curve corresponds to a variable in the order of the variables written in the function. λ , the penalizing factor, shows the path of its coefficient against log λ as λ varies. The number of nonzero coefficients at the current λ is indicated on the top axis, i.e. the effective degrees of freedom for the lasso. From left to right, it shows the number of nonzero coefficients. We see that a higher number of coefficients reach 0 as λ increases, hence being eliminated. Cross-validation was implemented to improve the model by selecting the best value of lambda at which accurate predictions are made. We then predict the ObamaPercentMargin.

The model did not prove to be the most accurate, possibly due to the lower number of features fed into it. The most significant variables were Black, Bachelors, HighSchool, White, MedianIncome and Homeowner.

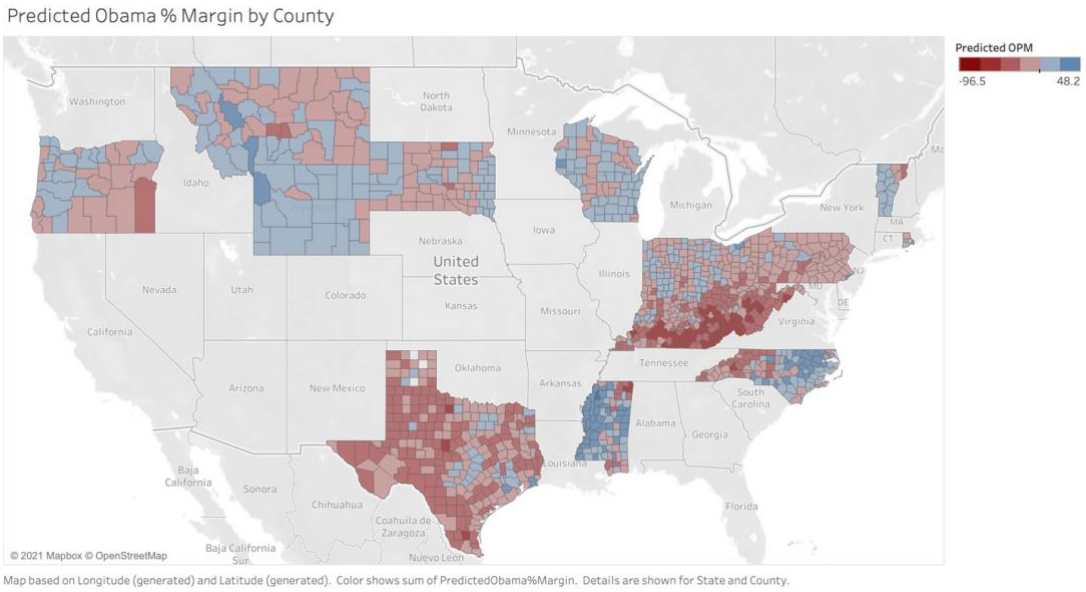


RMSE For All Prediction Models:

Model	RMSE on testing data
Linear regression	20.84
Random Forest	19.91
LASSO	22

The best prediction model proved to be the Random Forest with 83 trees and this model was used to map the unknown values for Obama Percent Margin in Tableau.

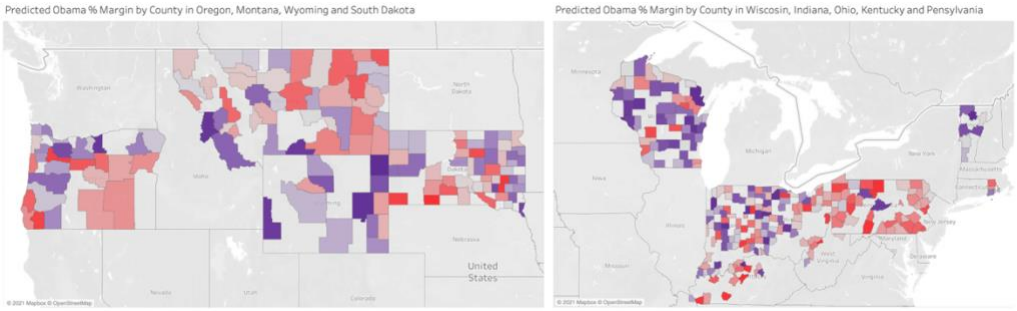
All Predicted OPM's by County

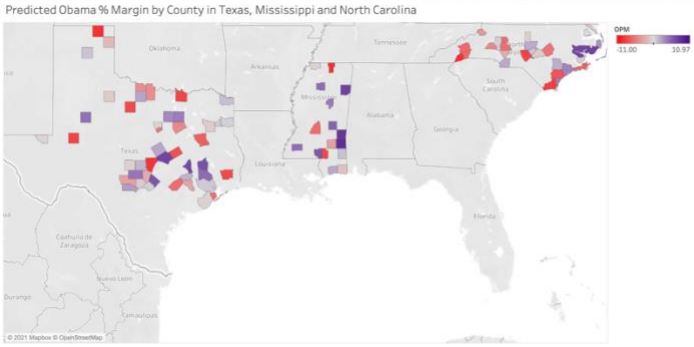


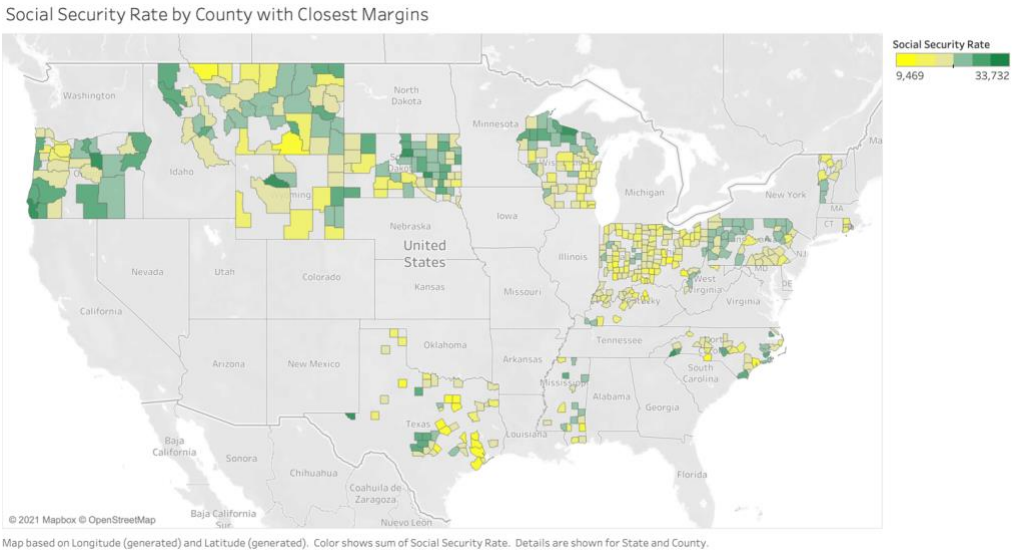
The states with the worst predicted OPMs are Texas, West Virginia, Kentucky and Pennsylvania and the states with the best predicted OPMs are Mississippi, Wyoming and North Carolina., and North Carolina.

Counties with the Closest Vote Margin

The maps below show the 500 counties with the closest vote margin and their social security rate.







Map based on Longitude (generated) and Latitude (generated). Color shows sum of Social Security Rate. Details are shown for State and County.

5) Conclusions and Recommendations

Among the focus attributes, Black, White, HighSchool, SocialSecurityRate and IncomeAbove75K generated the best prediction of OPM.

Thus, we recommend Obama to prioritize marketing in the non-voted counties with the:

- Closest Vote Margin
- Highest Black Population
- Lowest Social Security Rate
- Highest % High School Graduates

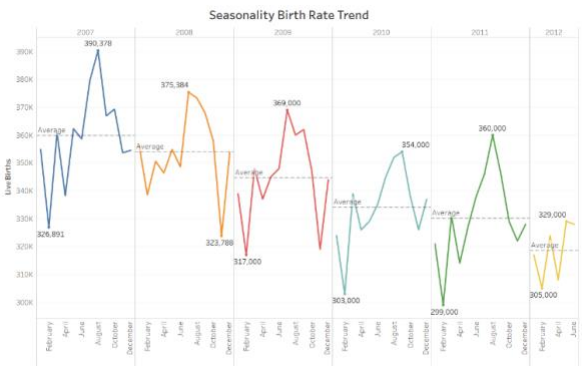
We have chosen to concentrate marketing the counties with the closest vote margin as Obama would be most likely to overtake or maintain a lead, rather than spending time and money in areas that he is extremely likely to win or lose either way. We recognized there are additional factors that differentiate counties even with the same vote margin, which is why we also considered the relevant attributes above in our prediction models. Marketing in these areas would allow, in theory, the same amount of marketing to produce greater results.

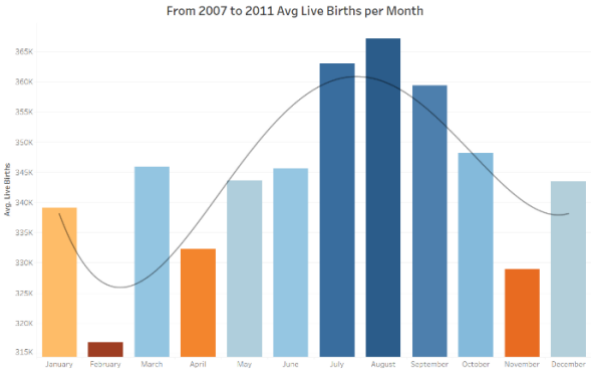
Out of the counties that have the closest vote margin, Obama should choose how many counties he wants to advertise in depending on the remaining time of his campaign and his available budget. Which counties come first should depend on their individual election dates.

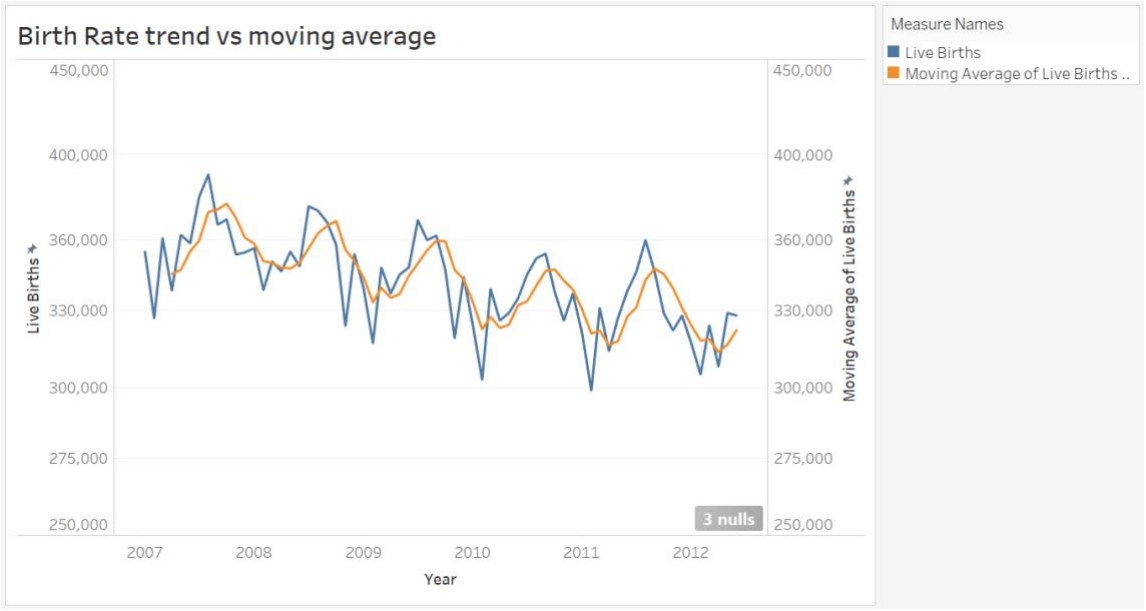
NICU Case Study

Patterns:

The seasonality on US birth shows that from June to October the number of births is above the average number of births for each year. Whereas February is when there is the lowest number of births in that year, except 2008. Moreover, overall, the trend of birth is decreasing, and we can know this as the average number of births for each year is declining.

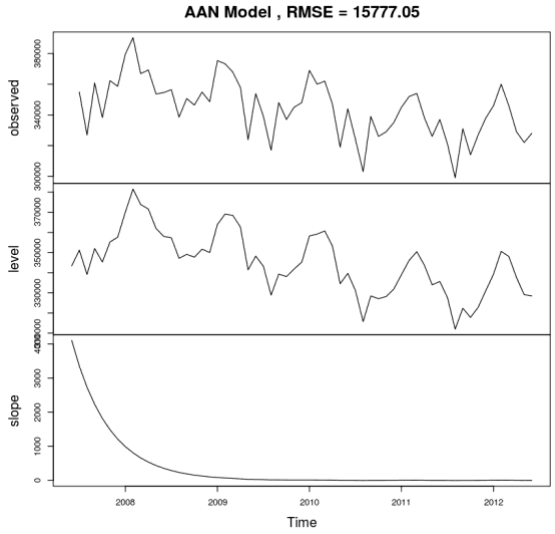


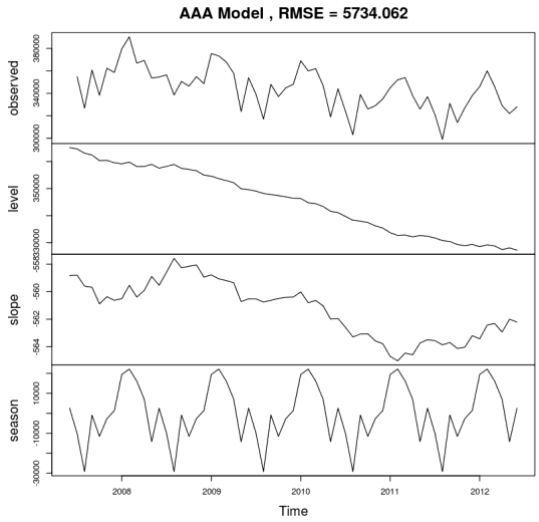


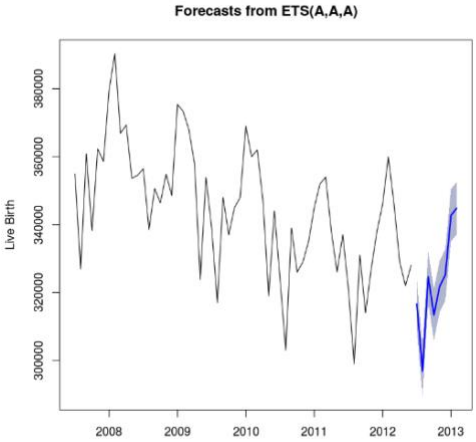


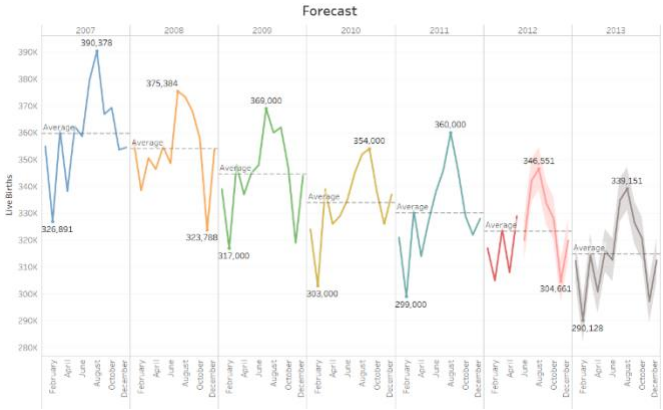
Model Prediction

The trend of data is analyzed by both AAN and AAA models. With RMSE of 5734, AAA is fitter than the AAN with RMSE of 15777. Therefore, the AAA model is used for forecasting the future trend of US births to February 2013 on both Tableau and R with an 80% confidence level.







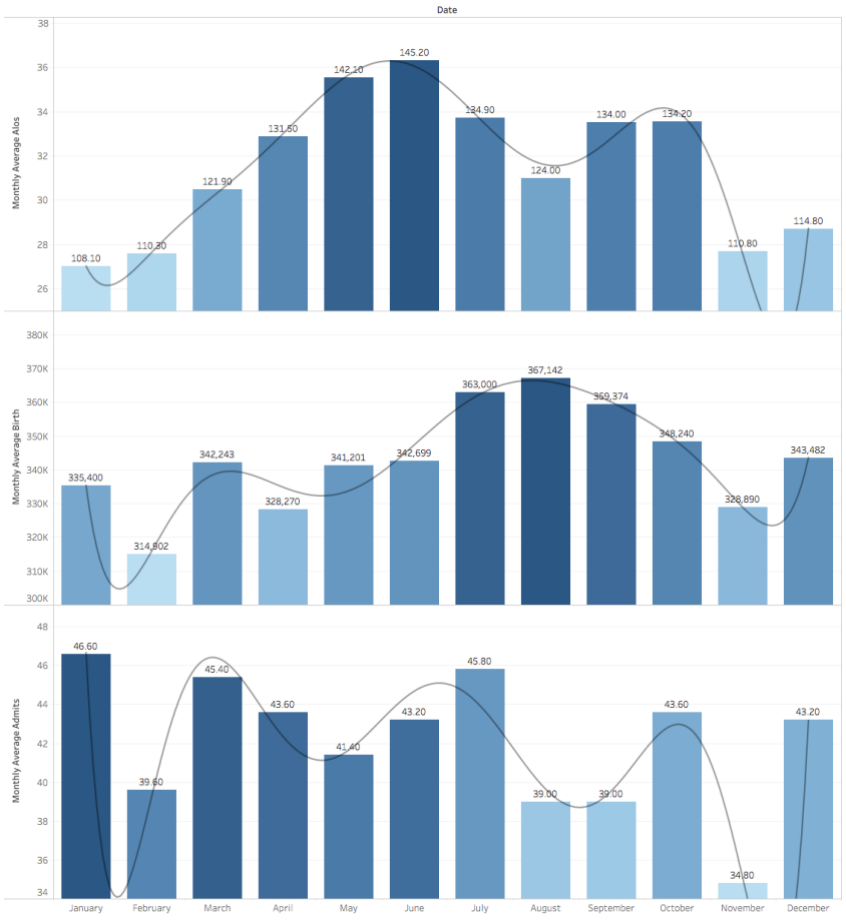


The birth rate has a general trend of the increase until 2012 August, followed by a trend of decrease until 2012 February.

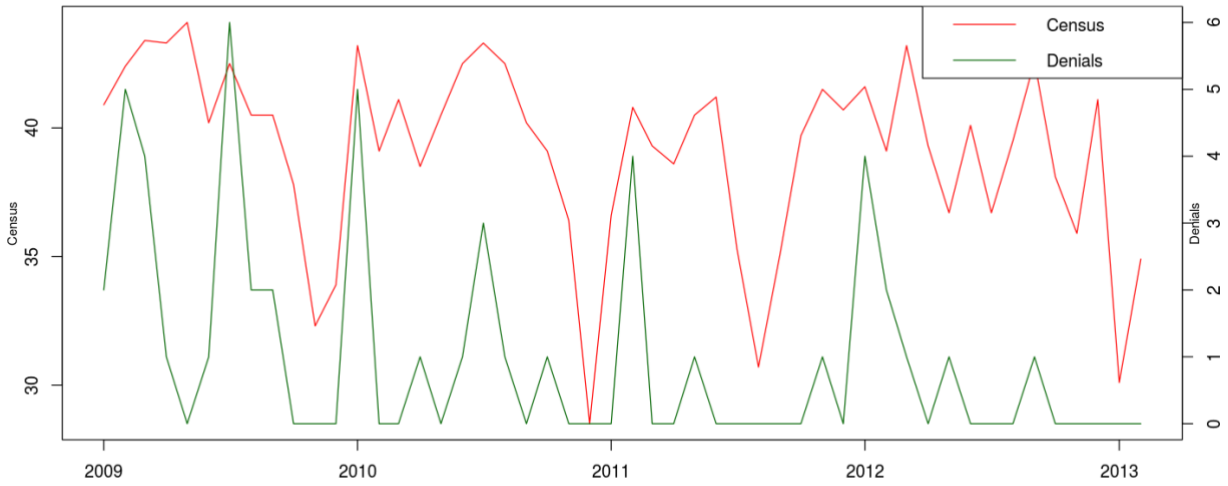
Comparison Analysis

For better analysis purposes, *NICU_A* and *US_Births* are merged in Tableau.

As showing above, from 2018 to 2012, the average live birth is increasing during the summer, with the highest record in August. A similar trend of increase is showing in the average admits in NICU, yet the maximum of admits is happening in January.



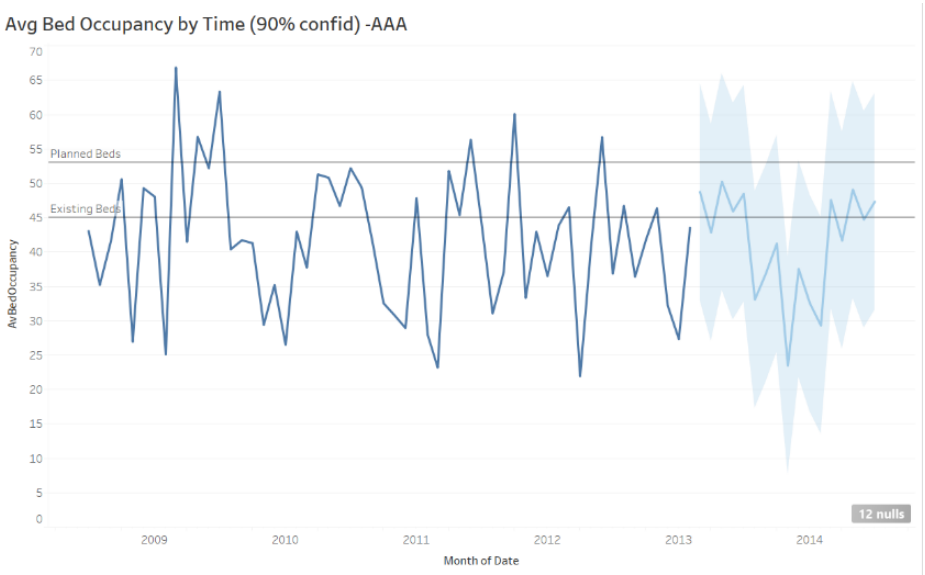
The peak can be explained by the seasonal trend of census and denials, as both census and denials are peaked at around January and August.



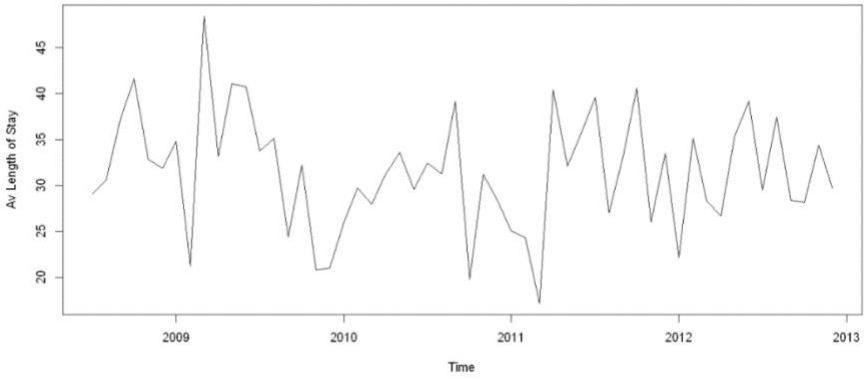
The difference between the maximum US birth rate and the admitted rate is mainly since there are not enough beds left to admit babies. In a month with a higher record of births, there is also a higher possibility of having a large number of premature babies, which would stay for a long time and occupy a lot of spaces making it unable to accept more babies in the next three months. This can be further proved by the minimum of ALOS that happened when there is the maximum admits in NICU.

Recommendation

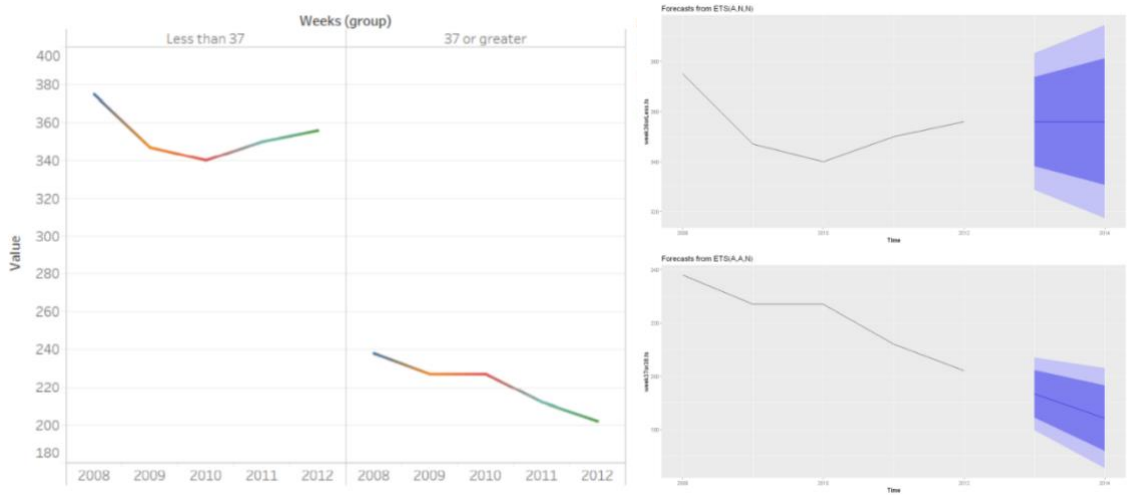
The Census data is the estimated average daily bed occupancy in each month, calculated through Census = Admits X ALOS X 12/365. The Census was forecasted using two different models: AAN and AAA. Yet, AAA had the lower RMSE, showing better accuracy compare to AAN. Therefore, applying AAA in prediction, it is showing a small proportion average bed occupancy for 2014 will exceed the existing beds in NICU, with the majority month of the forecasted average bed occupancy in 2014 below the existing beds.



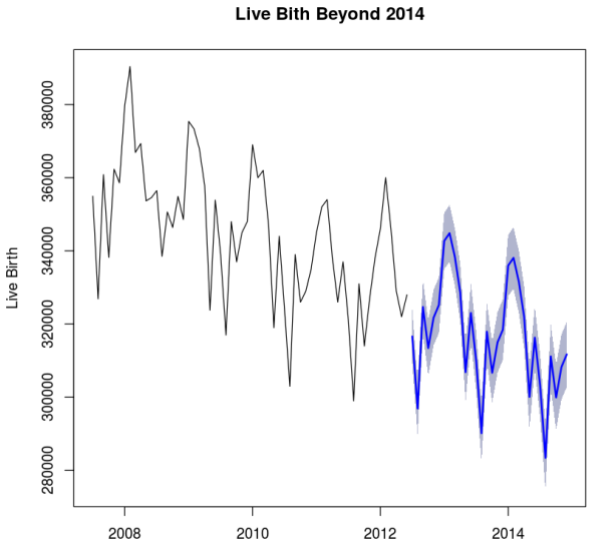
ALOS is also an important factor in determining the necessity of extra beds, it was decreasing from 2009 to 2010 and was having a slightly increasing trend from 2011 to 2013.



The trends in ALOS can be explained by the gestational data. The gestational data show a decreasing trend of babies born less than 37 months from 2008 to 2019, and the very opposite trend for the next following year, which is parallel to what is shown in ALOS. This makes sense as babies born before 37 weeks can have serious problems which would result in NICU admissions, so the more babies born in less than 37 weeks more are admitted into NICU. The predicted forecast of babies born before 37 weeks tends to have a straight line for 2014 and wouldn't be further increased. Thus, it would be safe to say it would not increase the ALOS in the next years.



Finally, another AAA prediction shows the live birth rate is predicted to continually decline beyond 2014.



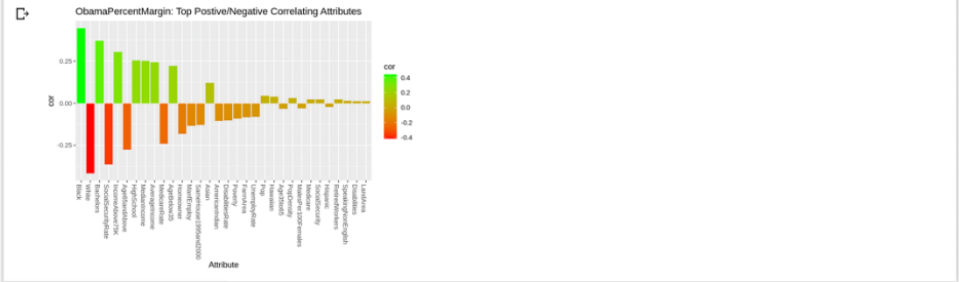
In conclusion, we suggest NICU to keep the existing 45 bed plan.

Code Appendix: Obama vs. Clinton Case Study

2) Understand the Data

Relevance of Attributes – R Barplot

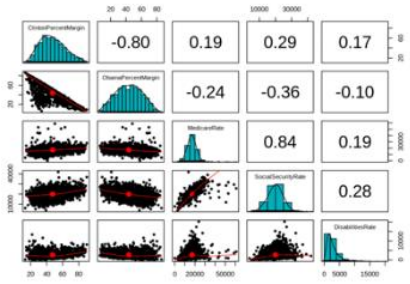
```r
options(repr.plot.height=5)

# Prepare
targetCol <- which(names(elect.df)=="ObamaPercentMargin")
startCol <- which(names(elect.df)=="MalesPer100Females")
endCol <- which(names(elect.df)=="FarmArea")

cor.ObamaPercentMargin <- cor(elect.df[,c(targetCol, startCol:endCol)]
                              ,use="complete.obs")[-1,"ObamaPercentMargin"]

cor.ObamaPercentMargin.df <- data.frame(cor=cor.ObamaPercentMargin, abs.cor=abs(cor.ObamaPercentMargin),
                                        row.names=names(cor.ObamaPercentMargin))

cor.ObamaPercentMargin.df <- cor.ObamaPercentMargin.df[order(-cor.ObamaPercentMargin.df$abs.cor),]

ggplot(cor.ObamaPercentMargin.df, aes(x=reorder(row.names(cor.ObamaPercentMargin.df),-abs.cor), y=cor, fill=cor)) +
  geom_col() + ggtitle("ObamaPercentMargin: Top Postive/Negative Correlating Attributes") + xlab("Attribute") +
  scale_fill_gradient(low="red", high="green") +
  theme(axis.text.x=element_text(angle=-90, hjust=0))
```



OPM by Social Welfare – R Pairwise Plot

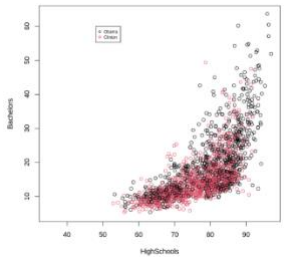
```r
[17]
library(dplyr)
install.packages("psych")
library(psych)

elect.df$ObamaPercentMargin <- (elect.df$Obama / elect.df$TotalVote)*100
elect.df$ClintonPercentMargin <- (elect.df$Clinton/elect.df$TotalVote) * 100
elect.df2 = elect.df %>% select(ClintonPercentMargin, ObamaPercentMargin,
                                MedicareRate, SocialSecurityRate,
                                DisabilitiesRate)
pairs.panels(elect.df2,
             method = "pearson", # correlation method
             hist.col = "#00AFBB",
             density = TRUE,  # show density plots
             ellipses = TRUE # show correlation ellipses
             )
```

```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)
```



Education by Margin Leader – R Scatterplot

```
[48] elect.df$leader = elect.df$Obama

     elect.df$leader[elect.df$Obama >= elect.df$Clinton]="Barack Obama"
     elect.df$leader[elect.df$Clinton > elect.df$Obama]="Clinton"
     elect.df$leader <- factor(elect.df$leader)
     plot(x=elect.df$HighSchool, y=elect.df$Bachelors,ylab="Bachelors", xlab="HighSchools",col=elect.df$leader)
     legend(x=48, y=60, c("Obama","Clinton"), cex =0.6, col=c("black","red"), pch=c(1))
```



Additional Graphs Not Included in Main Report:

Relevant Attributes – R Corellogram

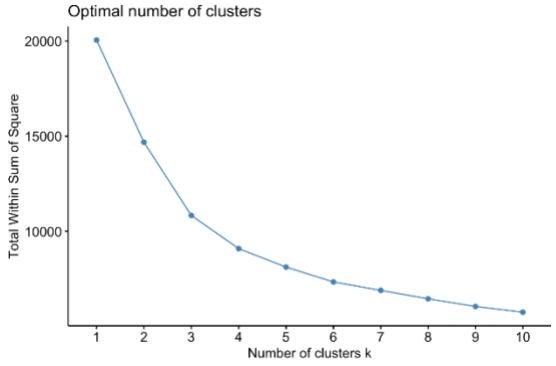
```
[49] install.packages("ellipse")
     library(ellipse)
     library(RColorBrewer)

     data=cor(elect.df[,c(11,12,13,14,15,20,21,22,24,28,30,35)],use="complete.obs")
     # Build a Pannel of 100 colors with Rcolor Brewer
     my_colors <- brewer.pal(5, "Spectral")
     my_colors=colorRampPalette(my_colors)(100)


     plotcorr(data , col=my_colors[data*50+50], mar = c(0,0,0,0),cex.lab=0.75 , type = "upper" , diag=FALSE)
```

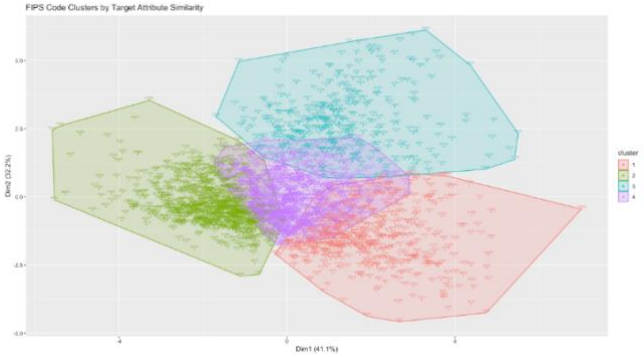
```
[49] Installing package into '/usr/local/lib/R/site-library'
     (as 'lib' is unspecified)
```

Fviz Cluster Plot

To explore similarities between FIPS codes by the relevant attributes, we initially created a K-means clustering plot. We chose 4 clusters according to the elbow method. While these clusters are not used in our prediction models, as predictions are formed on similarities between individual FIPS codes, this plot was to initially explore what types of demographics the dataset has.

```r
# K-MEANS CLUSTERING TO FIND SIMILAR FIPS BY TARGET ATTRIBUTES----

# Filter data to only include FIPS and target attributes
# Black, White, Education, Social Security, Medicare, Income, Age (below 35, above 65)
target_attributes <- c("FIPS",
                       "Black", "White",
                       "HighSchool", "Bachelors",
                       "SocialSecurityRate", "MedicareRate",
                       "Income",
                       "AgeBelow35", "AgeAbove65")

elect_targetatt <- elect.df[,names(elect.df) %in% target_attributes]

# Replace NA's with 0
elect_targetatt[is.na(elect_targetatt)] <- 0

# Make FIPS the rownames
elect_targetatt <- data.frame(elect_targetatt, row.names = 1)

# Scale the data (center it around 0)
elect_targetatt_scaled <- scale(elect_targetatt)

# Determine optimal number of clusters using "elbow" method
set.seed(123)
fviz_nbclust(elect_targetatt_scaled, kmeans, method = "wss")

# Perform clustering with 4 clusters
FIPS_kmeans <- kmeans(elect_targetatt_scaled, centers = 4, nstart = 20)
FIPS_kmeans

# Visualize k means clusters
FIPS_kmeans_plot = fviz_cluster(FIPS_kmeans, data = elect_targetatt_scaled,
                                main = "FIPS Code Clusters by Target Attribute Similarity",
                                labelsize = 5,
                                pointsize = 0.5)
FIPS_kmeans_plot
```





3) Prepare the Data

```r
install.packages("curl")
library(curl)
x <- curl("https://raw.githubusercontent.com/awhiter/TeachingRepos/master/datasets/Obama.csv")
elect.df  <- read.csv(x)
```

```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)
```

```r
# Create new derived target attributes

elect.df$ObamaRate <- 100 * elect.df$Obama / elect.df$TotalVote

elect.df$ClintonRate <- 100 * elect.df$Clinton / elect.df$TotalVote

elect.df$ObamaPercentMargin <- 100 * (elect.df$Obama - elect.df$Clinton) / elect.df$TotalVote

# Imputing missing values:
# Missing values for AverageIncome are replaced by the MedianIncome for that same record

elect.df$AverageIncome <- ifelse(is.na(elect.df$AverageIncome), elect.df$MedianIncome, elect.df$AverageIncome)

# Missing values for the following list of attributes are replaced by 0.

for (attr in c("Black","Asian","AmericanIndian","ManfEmploy","Disabilities","DisabilitiesRate","FarmArea"))
{elect.df[[attr]] <- ifelse(is.na(elect.df[[attr]]), 0, elect.df[[attr]])}

# The following codes removes all the remaining missing values
elect.df <- elect.df[is.na(elect.df$HighSchool)==FALSE,]
elect.df <- elect.df[is.na(elect.df$Poverty)==FALSE,]

# Check if all the missing data has addressed except for the missing vote data.
countNAs <- function (v) sum(ifelse(is.na(v),1,0))

elect.countNAs <- sapply(elect.df, countNAs)
elect.countNAs[elect.countNAs != 0]
```

TotalVote: 1130 Clinton: 1130 Obama: 1130 ObamaRate: 1130 ClintonRate: 1130 ObamaPercentMargin: 1130

```r
# Convert the ElectionDate column to the 'Date' data type
elect.df$ElectionDate <- as.Date(elect.df$ElectionDate, format="%m/%d/%Y")

# Create "known" and "unknown" vote data sets
elect.df.known <- elect.df[elect.df$ElectionDate < as.Date('2/19/2008', format = "%m/%d/%Y"), ]

elect.df.unknown <- elect.df[elect.df$ElectionDate >= as.Date('2/19/2008', format = "%m/%d/%Y"), ]

# Count the number of rows in our known and unknown datasets
nrow(elect.df.known)
nrow(elect.df.unknown)
```

1736
1130

```r
# Split the "known" vote data into separate training and test datasets
nKnown <- nrow(elect.df.known) # Find the number of rows in the known dataset

set.seed(201) # Set the seed for a random sample

rowIndicesTrain <- sample(1:nKnown, size = round(nKnown*0.75), replace = FALSE) # Randomly sample 75% of the row indices in the known dataset

# Split the training set into the training set and the test set using these indices.

elect.df.training <- elect.df.known[rowIndicesTrain, ]

elect.df.test <- elect.df.known[-rowIndicesTrain, ]
```

```r
head(elect.df.training)
summary(elect.df.training)
```

A data frame: 6 × 44

	County	State	Region	FIPS	ElectionDate	ElectionType	TotalVote	Clinton	Obama	MalesPer100Females	⋯	DisabilitiesRate	Homeowner	SameHouse19
	<chr>	<chr>	<chr>	<int>	<date>	<chr>	<int>	<int>	<int>	<dbl>	⋯	<dbl>	<dbl>	
1236	Texas	OK	South	40139	2008-02-05	Primary	541	301	260	107.2	⋯	850	68.8	
695	Blaine	ID	West	16013	2008-02-05	Primary	1568	336	1177	108.4	⋯	250	68.9	
1408	Red River	LA	South	22081	2008-02-09	Primary	871	369	517	90.9	⋯	5431	78.3	
555	Chattahoochee	GA	South	13053	2008-02-05	Primary	406	153	233	176.0	⋯	817	27.0	
177	Brevard	FL	South	12009	2008-01-29	Primary	62700	21713	18412	96.6	⋯	1601	74.6	
902	Norman	MN	Midwest	27107	2008-02-05	Caucuses	79	33	46	99.8	⋯	1842	81.0	

```
   County              State               Region                FIPS
 Length:1302        Length:1302        Length:1302        Min.   : 1001
 Class :character   Class :character   Class :character   1st Qu.:13152
 Mode  :character   Mode  :character   Mode  :character   Median :24046
                                                          Mean   :25716
                                                          3rd Qu.:36062
                                                          Max.   :53075
  ElectionDate        ElectionType          TotalVote           Clinton
 Min.   :2008-01-03   Length:1302        Min.   :     11.0   Min.   :     4.6
 1st Qu.:2008-02-05   Class :character   1st Qu.:    701.5   1st Qu.:   315.2
 Median :2008-02-05   Mode  :character   Median :   2310.0   Median :  1083.5
 Mean   :2008-02-03                      Mean   :  12425.7   Mean   :  5841.9
 3rd Qu.:2008-02-05                      3rd Qu.:   6820.5   3rd Qu.:  3221.6
 Max.   :2008-02-12                      Max.   :325188.0    Max.   :152817.0
     Obama          MalesPer100Females   AgeBelow35      Age35to65
 Min.   :     4.0   Min.   : 76.20    Min.   :29.10   Min.   :18.60
 1st Qu.:   255.5   1st Qu.: 94.83    1st Qu.:42.12   1st Qu.:38.10
 Median :   890.5   Median : 97.55    Median :45.45   Median :39.60
 Mean   :  5563.9   Mean   : 99.26    Mean   :45.72   Mean   :39.53
 3rd Qu.:  2848.5   3rd Qu.:100.70    3rd Qu.:48.80   3rd Qu.:41.30
 Max.   :170509.0   Max.   :200.90    Max.   :75.80   Max.   :56.50
 Age65andAbove       White           Black            Asian
 Min.   : 2.20   Min.   :15.70   Min.   : 0.00   Min.   : 0.000
 1st Qu.:12.30   1st Qu.:78.30   1st Qu.: 0.60   1st Qu.: 0.300
 Median :14.30   Median :91.90   Median : 3.20   Median : 0.500
 Mean   :14.75   Mean   :85.45   Mean   :10.99   Mean   : 1.245
 3rd Qu.:17.30   3rd Qu.:97.20   3rd Qu.:15.38   3rd Qu.: 1.100
 Max.   :33.70   Max.   :99.80   Max.   :82.80   Max.   :32.900
```

```
 AmericanIndian      Hawaiian            Hispanic         HighSchool
 Min.   : 0.000   Min.   :0.00000   Min.   : 0.100   Min.   :53.30
 1st Qu.: 0.200   1st Qu.:0.00000   1st Qu.: 1.300   1st Qu.:[illegible]
 Median : 0.400   Median :0.00000   Median : 2.600   Median :79.20
 Mean   : [illegible]   Mean   :0.06077   Mean   : [illegible]   Mean   :77.49
 3rd Qu.: 0.800   3rd Qu.:0.10000   3rd Qu.: 6.375   3rd Qu.:84.28
 Max.   :[illegible]   Max.   :1.10000   Max.   :[illegible]   Max.   :96.20
   Bachelors         Poverty          IncomeAbove75K     MedianIncome
 Min.   : 3.40   Min.   : 2.60   Min.   : [illegible]   Min.   :21920
 1st Qu.:11.48   1st Qu.: 9.70   1st Qu.: 9.225   1st Qu.:32753
 Median :14.80   Median :12.70   Median :[illegible]   Median :[illegible]
 Mean   :17.18   Mean   :13.35   Mean   :14.730   Mean   :40261
 3rd Qu.:19.90   3rd Qu.:16.60   3rd Qu.:17.275   3rd Qu.:44750
 Max.   :[illegible]   Max.   :36.00   Max.   :[illegible]   Max.   :99245
 AverageIncome    UnemployRate       ManfEmploy      SpeakingNonEnglish
 Min.   : [illegible]   Min.   : 1.900   Min.   : 0.000   Min.   : [illegible]
 1st Qu.:23208   1st Qu.: 3.600   1st Qu.: 2.404   1st Qu.: 2.927
 Median :26246   Median : 4.300   Median : 7.763   Median : 4.500
 Mean   :[illegible]   Mean   : 4.621   Mean   : [illegible]   Mean   : 7.85
 3rd Qu.:[illegible]   3rd Qu.: 5.400   3rd Qu.:[illegible]   3rd Qu.: 8.300
 Max.   :93377   Max.   :15.300   Max.   :57.229   Max.   :68.600
    Medicare       MedicareRate    SocialSecurity   SocialSecurityRate
 Min.   :     7   Min.   :    60   Min.   :    85   Min.   : 3883
 1st Qu.:  2273   1st Qu.:13907   1st Qu.:  2431   1st Qu.:16542
 Median :  4426   Median :16820   Median :  5300   Median :20000
 Mean   : [illegible]   Mean   :17036   Mean   : [illegible]   Mean   :19865
 3rd Qu.: [illegible]   3rd Qu.:19948   3rd Qu.: [illegible]   3rd Qu.:23144
 Max.   :[illegible]   Max.   :58686   Max.   :[illegible]   Max.   :43943
 RetiredWorkers    Disabilities       DisabilitiesRate    Homeowner
 Min.   :    55   Min.   :     0.0   Min.   :    0   Min.   :19.40
 1st Qu.:  [illegible]   1st Qu.:   275.7   1st Qu.: 1117   1st Qu.:[illegible]
 Median :  3120   Median :   679.3   Median : 2142   Median :75.00
 Mean   : [illegible]   Mean   :  2706.2   Mean   : 2529   Mean   :73.62
 3rd Qu.:  [illegible]   3rd Qu.:  1663.2   3rd Qu.: 3392   3rd Qu.:78.70
 Max.   :[illegible]   Max.   :[illegible]   Max.   :[illegible]   Max.   :89.40
 SameHouse1995and2000        Pop               PopDensity           LandArea
 Min.   :[illegible]   Min.   :     [illegible]   Min.   :    0.23   Min.   :     2.0
 1st Qu.:53.10   1st Qu.:  [illegible]   1st Qu.:   26.53   1st Qu.:   439.2
 Median :58.60   Median :  [illegible]   Median :   45.47   Median :   621.0
 Mean   :57.87   Mean   : [illegible]   Mean   :  386.25   Mean   :  1082.7
 3rd Qu.:[illegible]   3rd Qu.:  [illegible]   3rd Qu.:  129.08   3rd Qu.:   939.0
 Max.   :[illegible]   Max.   :[illegible]   Max.   :[illegible]   Max.   :[illegible]
    FarmArea          ObamaRate        ClintonRate    ObamaPercentMargin
 Min.   :    0.0   Min.   : 6.25   Min.   :13.44   Min.   :-80.932
 1st Qu.:  [illegible]   1st Qu.:32.00   1st Qu.:[illegible]   1st Qu.:-[illegible]
 Median :  [illegible]   Median :44.59   Median :[illegible]   Median : -1.754
 Mean   :  [illegible]   Mean   :43.95   Mean   :47.13   Mean   :  3.187
 3rd Qu.: 325.8   3rd Qu.:56.20   3rd Qu.:57.66   3rd Qu.: 18.116
 Max.   :3777.0   Max.   :86.05   Max.   :88.57   Max.   : 72.610
```

4) Prediction Models

Linear Regression

Linear Regression

```
[ ]  #load packages
     install.packages("stats")
     install.packages("dplyr")

     #load libraries
     library(stats)
     library(dplyr)
```

```
     Installing package into '/usr/local/lib/R/site-library'
     (as 'lib' is unspecified)

     Warning message:
     "package 'stats' is a base package, and should not be updated"
     Installing package into '/usr/local/lib/R/site-library'
     (as 'lib' is unspecified)
```

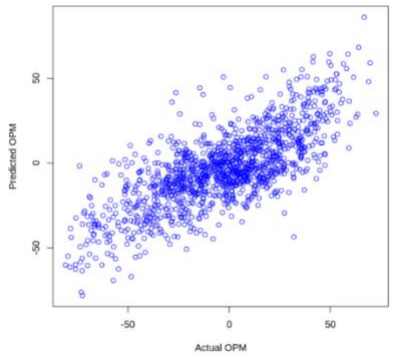
```
[50] LM_model = lm(ObamaPercentMargin~AverageIncome+MedianIncome+Black+White+HighSchool+Bachelors+MedicareRate+SocialSecurityRate+IncomeAbove75K+AgeBelow35
     summary(LM_model)
```

```
     Call:
     lm(formula = ObamaPercentMargin ~ AverageIncome + MedianIncome +
         Black + White + HighSchool + Bachelors + MedicareRate + SocialSecurityRate +
         IncomeAbove75K + AgeBelow35 + Age65andAbove, data = elect.df.training)

     Residuals:
         Min       1Q  Median      3Q     Max
     -67.856 -12.434   0.392  13.795  69.058
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-67.856 -12.434   0.392  13.795  69.058

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)       -1.429e+02  2.098e+01  -6.812 1.47e-11 ***
AverageIncome     -3.577e-04  1.534e-04  -2.331  0.01991 *
MedianIncome       7.997e-04  2.084e-04   3.838  0.00013 ***
Black              1.709e+00  1.184e-01  14.437  < 2e-16 ***
White              3.302e-01  1.201e-01   2.750  0.00604 **
HighSchool         1.624e+00  1.267e-01  12.815  < 2e-16 ***
Bachelors          6.950e-01  1.584e-01   4.387 1.24e-05 ***
MedicareRate       3.672e-04  2.225e-04   1.650  0.09912 .
SocialSecurityRate -2.380e-03  3.645e-04  -6.532 9.33e-11 ***
IncomeAbove75K     -1.447e+00  2.952e-01  -4.903 1.07e-06 ***
AgeBelow35         -4.176e-01  2.028e-01  -2.060  0.03964 *
Age65andAbove       9.434e-01  3.996e-01   2.361  0.01837 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 20.6 on 1290 degrees of freedom
Multiple R-squared:  0.5275,    Adjusted R-squared:  0.5235
F-statistic: 130.9 on 11 and 1290 DF,  p-value: < 2.2e-16
```

```
[40] # prediction and accuracy
     LM_model_rmse_train <- elect.df.training %>%
       mutate(pred.ObamaPercentMargin.train = predict(LM_model))
```

```
[ ]  plot(LM_model_rmse_train$ObamaPercentMargin, LM_model_rmse_train$pred.ObamaPercentMargin.train,xlab="Actual OPM", ylab="Predicted OPM", col=c("blue"))
```



```
[42] mse <- LM_model_rmse_train %>%
       mutate(error=pred.ObamaPercentMargin.train - ObamaPercentMargin,
              sq.error = error^2) %>%
       summarise(mse=mean(sq.error))
     rmse <- sqrt(mse)
     rmse
```

A data.frame: 1 × 1

mse
<dbl>
20.02024

```
[ ] #check rmse on test data
    pred.ObamaPercentMargin.test <- predict(LM_model, newdata = elect.df.test)
    rmse.test <- sqrt(mean((pred.ObamaPercentMargin.test - elect.df.test$ObamaPercentMargin)^2))
    rmse.test
```

20.8443191543856

Random Forest

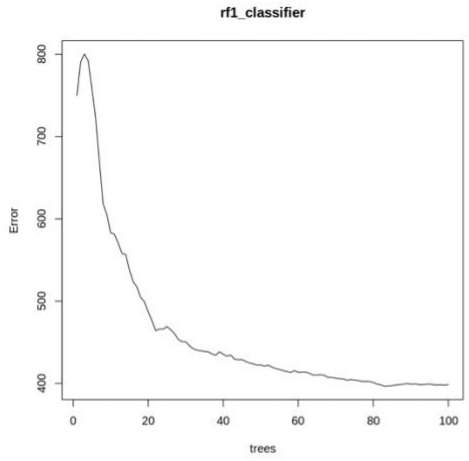
In [ ]:
```
install.packages("randomForest")
install.packages("ranger")
```

```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)

Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)

also installing the dependency 'RcppEigen'
```

In [ ]:
```
library(randomForest)
library(ranger)

rf1_classifier=randomForest(ObamaPercentMargin ~ Age65andAbove+White
                            +Black+HighSchool+Bachelors+IncomeAbove75K+MedicareRate+
                              SocialSecurityRate+MedianIncome+AverageIncome+AgeBelow35,
                           data = elect.df.test, ntree=100, mtry=5,
                           importance=TRUE)
plot(rf1_classifier)

# number of trees with lowest MSE
which.min(rf1_classifier$mse)

# RMSE of this optimal random forest
sqrt(rf1_classifier$mse[which.min(rf1_classifier$mse)])
```

83

19.9116347628852



In [ ]:
```
library(randomForest)
RandomForest_classifier=randomForest(ObamaPercentMargin ~ Age65andAbove+White
                            +Black+HighSchool+Bachelors+IncomeAbove75K+MedicareRate+
                              SocialSecurityRate+MedianIncome+AverageIncome+AgeBelow35,
                            data = elect.df.test, ntree=83, mtry=5,
                            importance=TRUE)

RandomForest_classifier
```

```
Call:
 randomForest(formula = ObamaPercentMargin ~ Age65andAbove + White +      Black + HighSchool + Bachelors + IncomeAbov
e75K + MedicareRate +      SocialSecurityRate + MedianIncome + AverageIncome + AgeBelow35,      data = elect.df.test,
ntree = 83, mtry = 5, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 83
No. of variables tried at each split: 5

          Mean of squared residuals: 393.2986
                    % Var explained: 60.72
```

In [ ]:
```
pred<-predict(RandomForest_classifier,elect.df.unknown) #Predictions on Test Set
```

Tuning

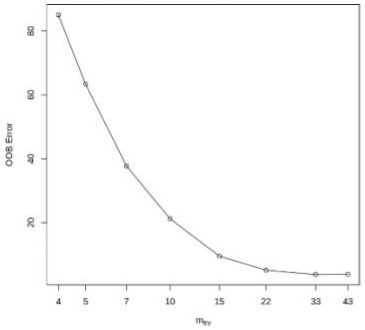
Initial tuning with randomForest

tuneRf starts at a value of mtry that we provided (mytryStart =5) and increase by a factor of 1.5 until the OOB error stops improving by 1%.

Type *Markdown* and LaTeX: α^2

In [ ]:
```
# names of features
features <- setdiff(names(elect.df.test), "ObamaPercentMargin")

set.seed(123)

rf_classifier2 <- tuneRF(
  x          = elect.df.test[features],
  y          = elect.df.test$ObamaPercentMargin,
  ntreeTry   = 500,
  mtryStart  = 5,
  stepFactor = 1.5,
  improve    = 0.01,
  trace      = FALSE) # to not show real-time progress
```

```
-0.3424022 0.01
0.4044564 0.01
0.4372122 0.01
0.5524074 0.01
0.4608531 0.01
0.2620479 0.01
-0.01786982 0.01
```



Full grid search with ranger

To perform a larger grid search across several hyperparameters we created a grid and loop through each hyperparameter combination and evaluated the model.

In [ ]:
```
# randomForest speed
system.time(
  Obama_randomForest <- randomForest(
    ObamaPercentMargin ~ Age65andAbove+White
    +Black+HighSchool+Bachelors+IncomeAbove75K+MedicareRate+
    SocialSecurityRate+MedianIncome+AverageIncome+AgeBelow35,
    data = elect.df.test,
    ntree = 500,
    mtry  = 5))

hyper_grid <- expand.grid(
  mtry       = seq(20, 30, by = 2),
  node_size  = seq(3, 9, by = 2),
  sampe_size = c(.55, .632, .70, .80),
  OOB_RMSE   = 0
)

# ranger speed
system.time(
  Obama_ranger <- ranger(
    formula   = ObamaPercentMargin ~ Age65andAbove+White
    +Black+HighSchool+Bachelors+IncomeAbove75K+MedicareRate+
    SocialSecurityRate+MedianIncome+AverageIncome+AgeBelow35,
    data      = elect.df.test,
    num.trees = 500,
    mtry      = 5)
  )
nrow(hyper_grid)
```

```
   user  system elapsed
  0.674   0.014   0.689

   user  system elapsed
  0.619   0.009   0.330
```

96

To perform the grid search, first we want to construct our grid of hyperparameters. We're going to search across 96 different models with varying mtry, minimum node size, and sample size.

We loop through each hyperparameter combination and apply 500 trees that were sufficient to achieve a stable error rate. The random number generator seed was set to consistently sample the same observations for each sample size and make clearer the impact that each change makes. Our top 10 performing models all have OOB RMSE values right around 19.6 and models with slighly larger sample sizes (80%) and deeper trees (7-9 observations in a terminal node) perform best. We get a full range of mtry values showing up in our top 10 models.

In [ ]:
```
install.packages("magrittr") # package installations are only needed the first time you use it
install.packages("dplyr")    # alternative installation of the %>%
```

In [ ]:
```
library(ranger)
library(magrittr) # needs to be run every time you start R and want to use %>%
library(dplyr)

for(i in 1:nrow(hyper_grid)) {

  # train model
  model <- ranger(
    formula   = ObamaPercentMargin ~ Age65andAbove+White
    +Black+HighSchool+Bachelors+IncomeAbove75K+MedicareRate+
    SocialSecurityRate+MedianIncome+AverageIncome+AgeBelow35,
    data      = elect.df.test,
    num.trees       = 500,
    mtry            = 5,
    min.node.size   = hyper_grid$node_size[i],
    sample.fraction = hyper_grid$sampe_size[i],
    seed            = 123
  )
  # add OOB error to grid
  hyper_grid$OOB_RMSE[i] <- sqrt(model$prediction.error)
}

hyper_grid %>%
  dplyr::arrange(OOB_RMSE) %>%
  head(10)
```

A data.frame: 10 × 4

	mtry	node_size	sampe_size	OOB_RMSE
	<dbl>	<dbl>	<dbl>	<dbl>
1	20	7	0.8	19.66411
2	22	7	0.8	19.66411
3	24	7	0.8	19.66411
4	26	7	0.8	19.66411
5	28	7	0.8	19.66411
6	30	7	0.8	19.66411
7	20	9	0.8	19.66457
8	22	9	0.8	19.66457
9	24	9	0.8	19.66457
10	26	9	0.8	19.66457

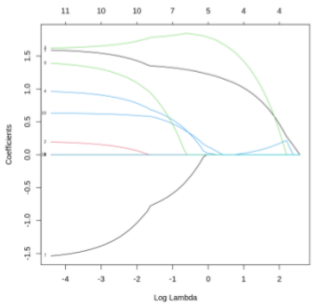
Currently, the best random forest model we have found retains columnar categorical variables and uses mtry = 20, terminal node size of 7 observations, and a sample size of 80%.

LASSO

Lasso Regularization

```
#set up input preparations
install.packages("glmnet", dependencies=TRUE)
library(glmnet)
grid = 10^seq(-5,0,length=100)
trainX = elect.df.training[,c("Black" ,"White" ,"HighSchool" ,"Bachelors", "SocialSecurityRate" , "MedianIncome" ,"IncomeAbove75K","AverageIncome",  "Age65andAbove", "Homeowner", "AgeBelow35" )]
y<- elect.df.training$ObamaPercentMargin
trainX = as.matrix(trainX)

#Generate the lasso model
lasso = glmnet(trainX,elect.df.training$ObamaPercentMargin)

#Plot LASSO
plot(lasso,xvar="lambda", label= TRUE)
```

```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)
```




```
[89] coef(lasso, s = exp(0))
```

```
12 x 1 sparse Matrix of class "dgCMatrix"
                              1
(Intercept)        -1.505168e+02
Black               1.225100e+00
White               .
HighSchool          1.785334e+00
Bachelors           2.434804e-02
SocialSecurityRate -7.071052e-04
MedianIncome        .
IncomeAbove75K      .
AverageIncome       .
Age65andAbove       .
Homeowner           1.197224e-01
AgeBelow35          .
```

Use Cross Validation to Optimise Lamba

```
[90] set.seed(101)

     lasso.cv <- cv.glmnet(trainX, y, nfolds = 5, family = "gaussian")

     #ookup the optimal lambda value (lambda.min) and also find the log of this value
     lasso.cv$lambda.min
     (minLogLambda <- log(lasso.cv$lambda.min))
```

```
0.0448374358468025
-3.10471186691813
```

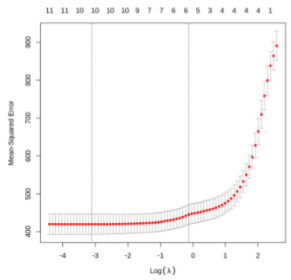
```
[91] coef(lasso.cv, s = "lambda.min")
     # Coefficients of the regularized linear regression with an optimal lambda.
```

```
12 x 1 sparse Matrix of class "dgCMatrix"
                              1
(Intercept)        -1.906541e+02
Black               1.548790e+00
White               1.594311e-01
HighSchool          1.648858e+00
Bachelors           9.141843e-01
SocialSecurityRate -2.383296e-03
MedianIncome        3.756635e-04
IncomeAbove75K     -1.406580e+00
AverageIncome       3.184594e-05
Age65andAbove       1.311658e+00
Homeowner           6.224027e-01
AgeBelow35          .
```

```
[92] log(lasso.cv$lambda.1se)
```

-0.127632153496426

```
[93] plot(lasso.cv)
```



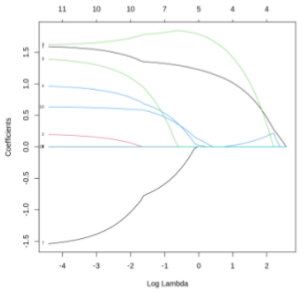
Prediction

```
[101] testX= elect.df.test[,c("Black" ,"White" ,"HighSchool" ,"Bachelors", "SocialSecurityRate" , "MedianIncome" ,"IncomeAbove75K","AverageIncome",  "Age65andAbove", "Homeowner", "AgeBelow35")]
      testX = as.matrix(testX)
```

```
[102] #set up input preparations
      install.packages("glmnet", dependencies=TRUE)
      library(glmnet)
      grid = 10^seq(-5,0,length=100)
      trainX = elect.df.training[,c("Black" ,"White" ,"HighSchool" ,"Bachelors", "SocialSecurityRate" , "MedianIncome" ,"IncomeAbove75K","AverageIncome",  "Age65andAbove", "Homeowner", "AgeBelow35" )]
      y<- elect.df.training$ObamaPercentMargin
      trainX = as.matrix(trainX)

      #Generate the lasso model
      lasso = glmnet(trainX,elect.df.training$ObamaPercentMargin)
```

```
[102] #Generate the lasso model
      lasso = glmnet(trainX,elect.df.training$ObamaPercentMargin)

      #Plot LASSO
      plot(lasso,xvar="lambda", label= TRUE)
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```



Calculating Error

```
[103] # first set library path to include the following directory
      # if running on Azure LinuxDataScience VM ...

      .libPaths('/home/vmuser/R/x86_64-pc-linux-gnu-library/3.2')

      # The metrics package includes the mae and rmse functions.
      # Install metrics if needed on Anaconda

      install.packages("Metrics")
      library(Metrics)
```

Calculating Error

```
[103] # first set library path to include the following directory
      # if running on Azure LinuxDataScience VM ...

      .libPaths('/home/vmuser/R/x86_64-pc-linux-gnu-library/3.2')

      # The metrics package includes the mae and rmse functions.
      # Install metrics if needed on Anaconda

      install.packages("Metrics")
      library(Metrics)

      genError <- function(prediction, actual)
          cat('MAE =', signif(mae(actual,prediction),4),
              ' RMSE =', signif(rmse(actual,prediction),4), "\n")
```

```
      Installing package into ‘/usr/local/lib/R/site-library’
      (as ‘lib’ is unspecified)
```

```
[104] lasso.cv.pred <- predict(lasso.cv, newx = testX, s = "lambda.min")

      genError(lasso.cv.pred, elect.df.test$ObamaPercentMargin)
```

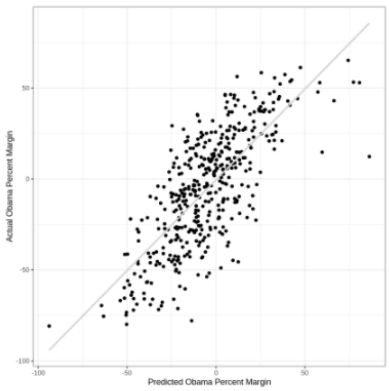
```
      MAE = 17.1  RMSE = 21.1
```

```
[105] #plotting lasso pregression vs actual values
      install.packages("cowplot")
      library(ggplot2)
      ggplot()  +
        geom_point(aes(lasso.cv.pred, elect.df.test$ObamaPercentMargin)) +
        geom_smooth(aes(lasso.cv.pred, elect.df.test$ObamaPercentMargin), method = "lm", se = FALSE, color = "lightgrey") +
        labs(x = "Predicted Obama Percent Margin", y = "Actual Obama Percent Margin") +
        theme_bw()
```

```
      Installing package into ‘/usr/local/lib/R/site-library’
      (as ‘lib’ is unspecified)

      `geom_smooth()` using formula 'y ~ x'
```

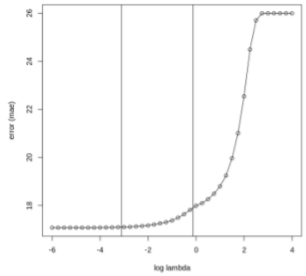
(as ‘lib’ is unspecified)

`geom_smooth()` using formula 'y ~ x'



Plot showing how the error (mae) varies with lambda

```
[106] errorvals<-sapply(seq(-6,4,0.25), function(loglambda){mae(predict(lasso.cv, newx = testX, s = exp(loglambda)),
                                elect.df.test$ObamaPercentMargin)})

      plot(seq(-6,4,0.25),errorvals, xlab="log lambda", ylab="error (mae)", type="o" )

      abline(v = log(lasso.cv$lambda.min))
      abline(v = log(lasso.cv$lambda.1se))
```




```
[107] coef(lasso.cv, s = "lambda.1se")
```

```
12 x 1 sparse Matrix of class "dgCMatrix"
                              1
(Intercept)        -1.546177e+02
Black               1.241698e+00
White               .
HighSchool          1.805956e+00
Bachelors           5.807934e-02
SocialSecurityRate -7.416551e-04
MedianIncome        .
IncomeAbove75K     -3.197912e-02
AverageIncome       .
Age65andAbove       .
Homeowner           1.591094e-01
AgeBelow35          .
```

```
[108] lasso.1se.pred <- predict(lasso.cv, newx = testX, s = "lambda.1se")

genError(lasso.1se.pred, elect.df.test$ObamaPercentMargin)
```

```
MAE = 17.93  RMSE = 22
```

Code Appendix: NICU Case Study

In [1]:
```
.libPaths("/usr/local/lib/R/site-library")
```

In [2]:
```
#downloading necessary library
library(caret)
library("forecast")
```

```
Loading required package: lattice
Loading required package: ggplot2
Registered S3 methods overwritten by 'ggplot2':
  method         from
  [.quosures     rlang
  c.quosures     rlang
  print.quosures rlang
Warning message:
"package 'forecast' was built under R version 3.6.3"Registered S3 method overwritten by 'xts':
  method     from
  as.zoo.xts zoo
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo
```

In [24]:
```
# default size of output plots
library(repr)
options(repr.plot.width=10, repr.plot.height=6)  # change plot size
```

Using Live Births Data

In [4]:
```
# Load Birth Dataframe
birth.df <- read.csv("US_Births.csv")
```

In [5]:
```
# View the Dataframe
n <- nrow(birth.df)
birth.df[c(1:4,(n-3):n),]
```

	Yr_Mo	Live.Births
1	200701	354943
2	200702	326891
3	200703	360828
4	200704	338224
63	201203	324000
64	201204	308000
65	201205	329000
66	201206	328000

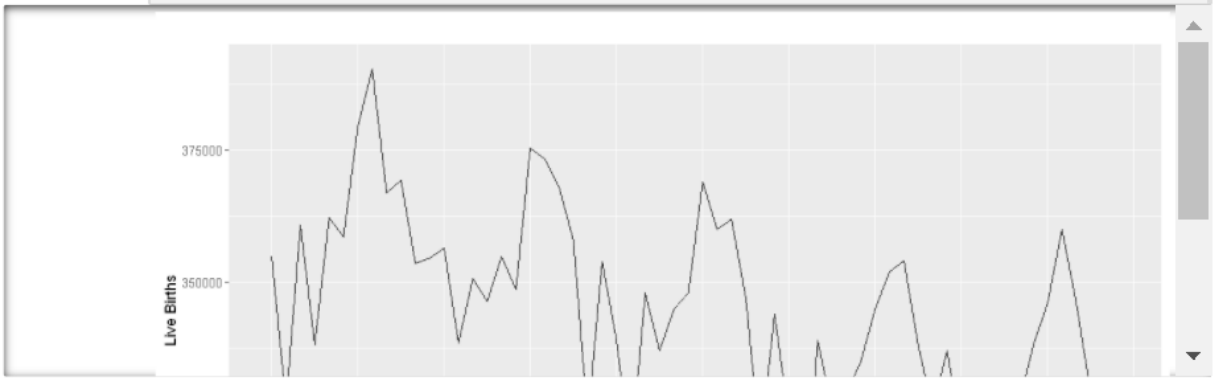
In [6]:

```
# Create Time Series
birth.ts <- ts(birth.df$Live.Births,
               start = c(2007, 7),
               end = c(2012, 6),
               freq = 12)
```

Seasonal Pattern of Live Births

In [7]:

```
autoplot(birth.ts, ylab = "Live Births")
```



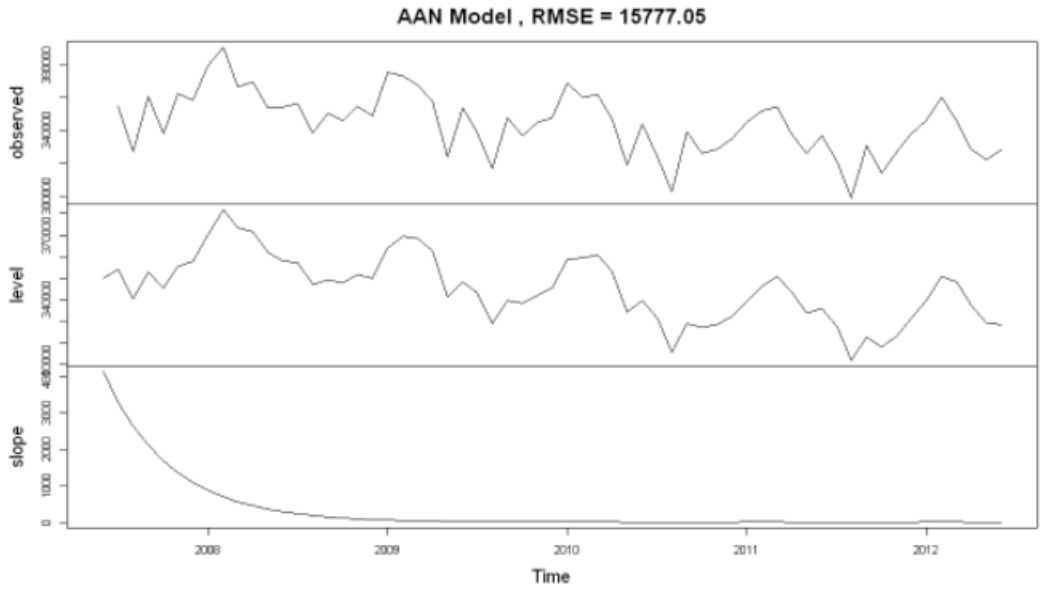
AAN Model

In [8]:
```
rmse.ets <- function (etsmodel) cat("RMSE = ",
                                    sqrt(etsmodel$mse))
```

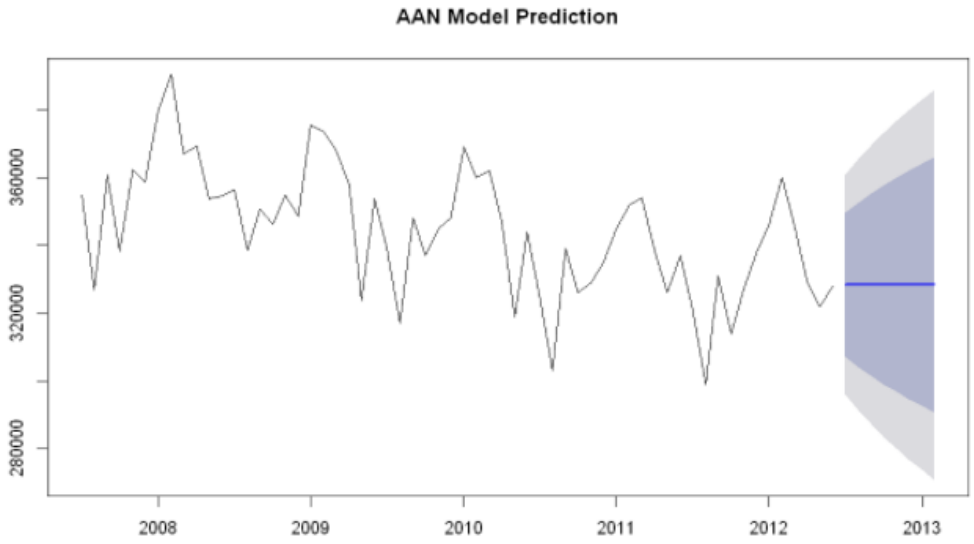
In [9]:
```
(Live.Births.ets.AAN <- ets(birth.ts, model = "AAN"))

rmse.ets(Live.Births.ets.AAN)
```

```
ETS(A,Ad,N)

Call:
 ets(y = birth.ts, model = "AAN")

  Smoothing parameters:
    alpha = 0.5547
    beta  = 1e-04
    phi   = 0.8

  Initial states:
    l = 349759.6135
    b = 4107.3425

  sigma:  16509.95

     AIC     AICc      BIC
1417.846 1419.431 1430.412

RMSE =  15807.07
```

In [10]:
```
par(col.main='white')
plot(Live.Births.ets.AAN)
par(col.main='black')
title("AAN Model , RMSE = 15777.05")
```



In [11]:
```
Live.Births.ets.AAN.pred <- forecast(Live.Births.ets.AAN, h = 8)
par(col.main='white')
plot(Live.Births.ets.AAN.pred)
par(col.main='black')
title("AAN Model Prediction")
```



In [12]:
```
Live.Births.ets.AAN.pred$mean[8]
```

328404.03024302

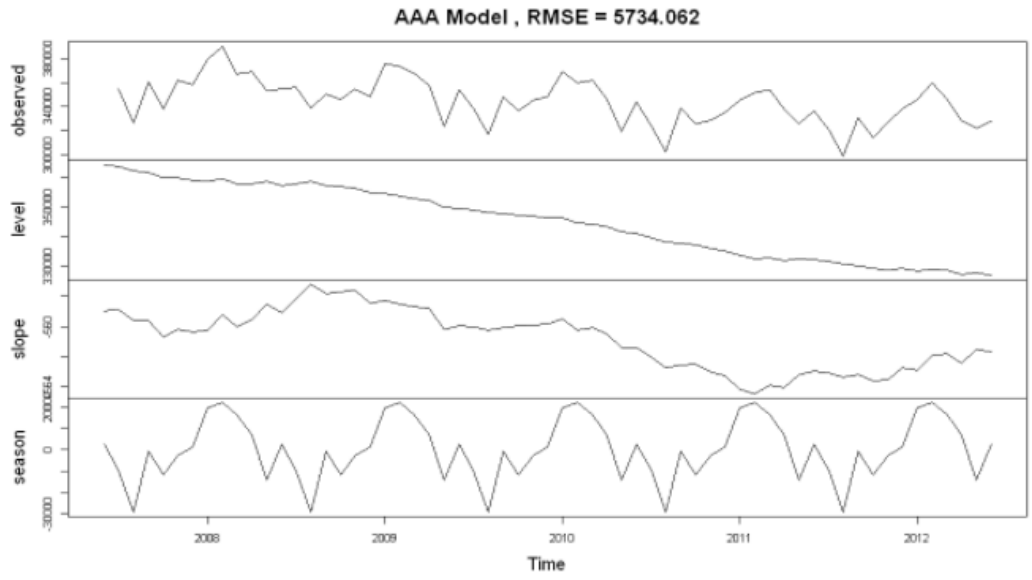
.

AAA Model

In [13]:
```
Live.Births.ets.aaa <- ets(birth.ts, model = "AAA")
rmse.ets(Live.Births.ets.aaa)
```

RMSE = 5718.357

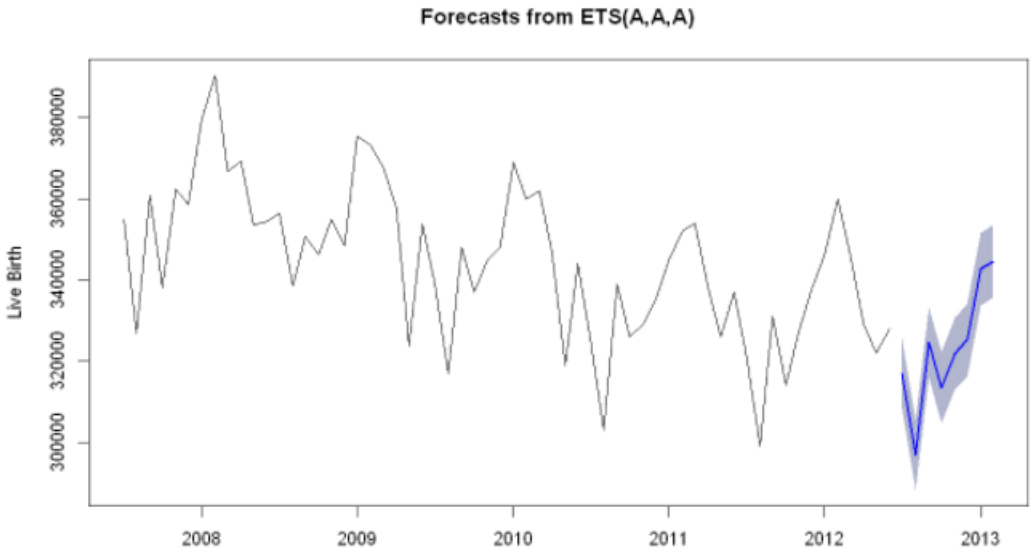
In [14]:
```
par(col.main='white')
plot(Live.Births.ets.aaa)
par(col.main='black')
title("AAA Model , RMSE = 5734.062")
```



In [15]:

```
cat('Feb 2013: live birth = ',round(forecast$mean[8],1),"\n")
cat('           upper 80% confid. live birth = ',round(forecast$upper[8,1],1),"\n")


plot(forecast, ylab = "Live Birth")
```

```
Feb 2013: live birth =  344592.3
          upper 80% confid. live birth =  353599.7
```



Birth AAA Prediction 2014

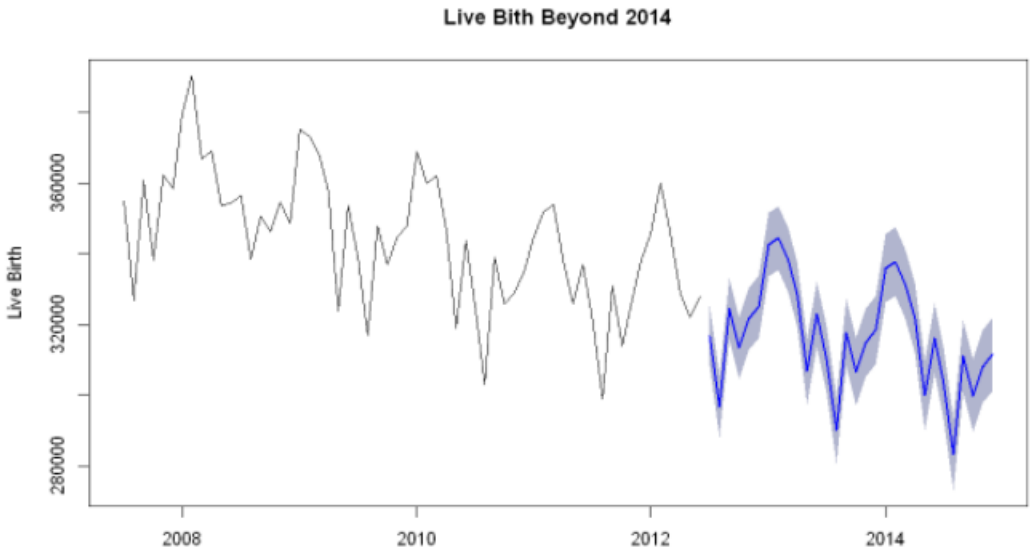
In [16]:
```
rmse.ets <- function (etsmodel) cat("RMSE = ",
                                    sqrt(etsmodel$mse))
```

In [17]:
```
Live.Births.ets.aaa <- ets(birth.ts, model = "AAA")
rmse.ets(Live.Births.ets.aaa)
```

```
RMSE =  5718.357
```

In [18]:
```
forecast <- forecast(Live.Births.ets.aaa, h = 30, level = c(80))

cat('July 2014: live birth = ',round(forecast$mean[8],1),"\n")
cat('           upper 80% confid. live birth = ',round(forecast$upper[8,1],1),"\n")

par(col.main='white')
plot(forecast, ylab = "Live Birth")
par(col.main='black')
title("Live Bith Beyond 2014")
```

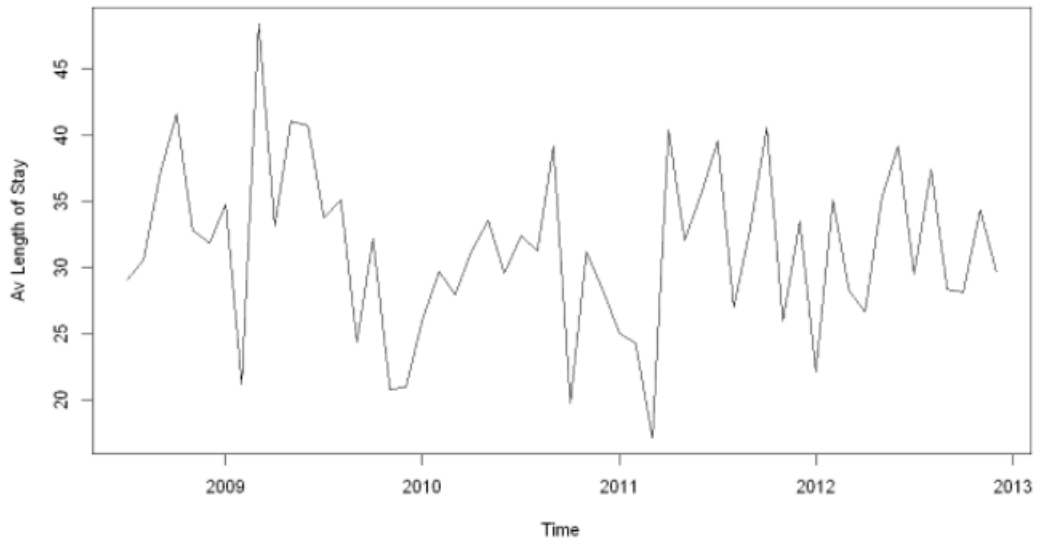
```
July 2014: live birth =  344592.3
           upper 80% confid. live birth =  353599.7
```



Using NICU Data

In [19]:
```
# Load NICU Dataframe
baby.df <- read.csv("NICU_A (1).csv")
```

In [20]:
```
# need to remove first 12 data values
alos.ts <- ts(baby.df$ALOS[-(1:12)],
              start = c(2008,7),
              end = c(2012,12),
              freq = 12)

plot(alos.ts, ylab = "Av Length of Stay")
```

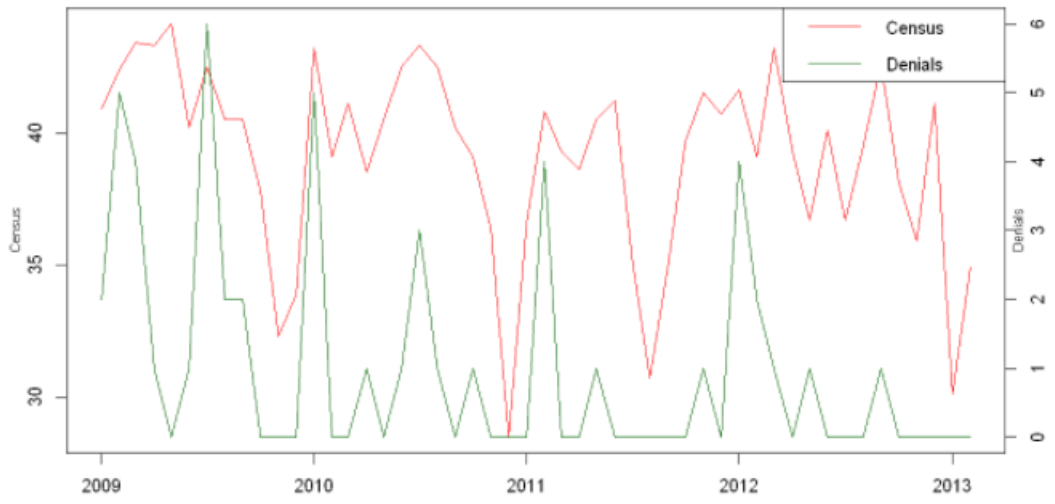


Using Census & Denial

In [21]:
```
#Load Census Dataframe
c_d.df <- read.csv("Census_Denials.csv")
```

In [22]:
```
c_d.df$Date <- as.Date(paste(as.character(c_d.df$Yr_Mo), "1",sep=""),
                       format="%Y%m%d")
```

In [23]:
```
plot(c_d.df$Date, c_d.df$Avg_Daily_Census, type="l", col="red",
     xlab=NA, ylab=NA)

par(new = T)
plot(c_d.df$Date, c_d.df$Denials, type="l", col="darkgreen",
     axes=F, xlab=NA, ylab=NA)

axis(side = 4)
mtext(side = 4, line = 0, 'Denials', cex=0.75)

mtext(side = 2, line = 2, 'Census', cex=0.75)

legend("topright",
       legend=c("Census", "","Denials"),
       col=c("red", "white","darkgreen"),lty=c(1,1,1))
```



Using Gestational Age

In [24]:
```
#Load Gestational Age Dataframe
gest.age <- read.csv("Gestational_Age.csv")
```

In [25]:
```
#Organize the data
rownames(gest.age)<-gest.age$Weeks
gest.age<-t(gest.age[,-1])
rownames(gest.age) <- 2008:2012
(gest.age.2cols <- data.frame(week36orLess=rowSums(gest.age[,1:15]), week37or38=gest.age[,16]))
```

	week36orLess	week37or38
2008	375	238
2009	347	227
2010	340	227
2011	350	212
2012	356	202

In [26]:
```
week36orLess.ts <- ts(gest.age.2cols$week36orLess, start=2008, end=2012, freq=1)

week36orLess.ts.model <- forecast(week36orLess.ts, h=3)
week36orLess<-autoplot(week36orLess.ts.model)
```

In [27]:
```
week37or38.ts <- ts(gest.age.2cols$week37or38, start=2008, end=2012, freq=1)
week37or38.ts.model <- forecast(week37or38.ts,h=3)
week37or38<-autoplot(week37or38.ts.model)
```

In [28]:
```
par(mfrow = c(2, 1))
plot(week36orLess)
plot(week37or38)
```