1992 U.S. Presidential election

Ali Tarek Maher Ibrahim Ali Seada and Paul Lovis Maximilian Trüstedt

24 6 2021

Read the data into R environment

```
library(pacman)
p_load(ggplot2,   # reportable graphs
       cowplot,   # arranges ggplot graphs nicely
       stargazer, # nice tables
       glmnet, # for regularization (lasso, ridge, elastic net)
       caret,      #  splitting the data and more
       rpart,      #  building decision trees
       rpart.plot,
       pROC)      # ROC AUC
rm(list=ls())
vote<-read.csv("vote92.csv", sep = ",", header = T,stringsAsFactors = T)
str(vote)
```

```
## 'data.frame':    909 obs. of  10 variables:
##  $ X          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ vote       : Factor w/ 3 levels "Bush","Clinton",..: 1 1 2 1 2 2 3 1 1 3 ...
##  $ dem        : int  0 0 1 0 0 1 1 0 0 0 ...
##  $ rep        : int  1 1 0 1 0 0 0 1 1 1 ...
##  $ female     : int  1 1 1 0 1 1 1 0 1 0 ...
##  $ persfinance: int  1 0 0 0 0 -1 1 0 1 0 ...
##  $ natlecon   : int  0 -1 -1 -1 -1 -1 0 0 -1 0 ...
##  $ clintondis : num  4.0804 4.0804 1.0404 0.0004 0.9604 ...
##  $ bushdis    : num  0.102 0.102 1.742 5.382 11.022 ...
##  $ perotdis   : num  0.26 0.26 0.24 2.22 6.2 ...
```

```
summary(vote)
```

```
##        X            vote          dem              rep             female
##  Min.   :  1   Bush   :310   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:228   Clinton:416   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :455   Perot  :183   Median :0.0000   Median :0.0000   Median :0.0000
##  Mean   :455                 Mean   :0.4884   Mean   :0.4301   Mean   :0.4752
##  3rd Qu.:682                 3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :909                 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##   persfinance           natlecon          clintondis          bushdis
##  Min.   :-1.000000   Min.   :-1.0000   Min.   : 0.0004   Min.   : 0.1024
##  1st Qu.:-1.000000   1st Qu.:-1.0000   1st Qu.: 0.9604   1st Qu.: 0.4624
```

```
##  Median : 0.000000   Median :-1.0000   Median : 1.0404   Median : 1.7424
##  Mean   :-0.009901   Mean   :-0.6722   Mean   : 3.5062   Mean   : 3.3793
##  3rd Qu.: 1.000000   3rd Qu.: 0.0000   3rd Qu.: 4.0804   3rd Qu.: 5.3824
##  Max.   : 1.000000   Max.   : 1.0000   Max.   :16.1600   Max.   :18.6620
##     perotdis
##  Min.   : 0.2401
##  1st Qu.: 0.2401
##  Median : 2.2201
##  Mean   : 2.1710
##  3rd Qu.: 2.2801
##  Max.   :12.1800
```

```
# ??remove cowplot, stargazer & pROC
```

Preprocess the data, preparing it for the modeling

```
vote$vote_num <- as.numeric(vote$vote)
vote$dem<-as.factor(vote$dem)
vote$rep<-as.factor(vote$rep)
vote$female<-as.factor(vote$female)
vote$persfinance<-as.factor(vote$persfinance)
vote$natlecon<-as.factor(vote$natlecon)
vote$polID<-as.factor((as.numeric(vote$dem)-1)+(as.numeric(vote$rep)*2-1))
str(vote)
```

```
## 'data.frame':    909 obs. of  12 variables:
##  $ X          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ vote       : Factor w/ 3 levels "Bush","Clinton",..: 1 1 2 1 2 2 3 1 1 3 ...
##  $ dem        : Factor w/ 2 levels "0","1": 1 1 2 1 1 2 2 1 1 1 ...
##  $ rep        : Factor w/ 2 levels "0","1": 2 2 1 2 1 1 1 2 2 2 ...
##  $ female     : Factor w/ 2 levels "0","1": 2 2 2 1 2 2 2 1 2 1 ...
##  $ persfinance: Factor w/ 3 levels "-1","0","1": 3 2 2 2 2 1 3 2 3 2 ...
##  $ natlecon   : Factor w/ 3 levels "-1","0","1": 2 1 1 1 1 1 2 2 1 2 ...
##  $ clintondis : num  4.0804 4.0804 1.0404 0.0004 0.9604 ...
##  $ bushdis    : num  0.102 0.102 1.742 5.382 11.022 ...
##  $ perotdis   : num  0.26 0.26 0.24 2.22 6.2 ...
##  $ vote_num   : num  1 1 2 1 2 2 3 1 1 3 ...
##  $ polID      : Factor w/ 3 levels "1","2","3": 3 3 2 3 1 2 2 3 3 3 ...
```

We decided to change some of the numeric variables to factors, because it makes more sense to have them as categorical than as numeric variables. Also this way, we can see, that there are no problems with the categorical variables regarding wrong values, because all provided levels are described by the given data set definition. Additionally we create a categorical variable called polID to summarize which political party the respondent is identifying himself with.

- treat missing values

```
colSums(is.na(vote))
```

```
##           X        vote         dem         rep      female persfinance
```

```
##           0           0           0           0           0           0
##    natlecon  clintondis     bushdis    perotdis    vote_num       polID
##           0           0           0           0           0           0
```

There are no missing values in this data set. No NAs, as well as data, that could otherwise be identified as missing.

- handle sparse classes of categorical predictors

```
table(vote$vote) # !!make these tables pretty (bar plot coloured)
```

```
##
##    Bush Clinton   Perot
##     310     416     183
```

```
table(vote$dem)
```

```
##
##   0   1
## 465 444
```

```
table(vote$rep)
```

```
##
##   0   1
## 518 391
```

```
table(vote$female)
```

```
##
##   0   1
## 477 432
```

```
table(vote$persfinance)
```

```
##
##  -1   0   1
## 308 302 299
```

```
table(vote$natlecon)
```

```
##
##  -1   0   1
## 656 208  45
```

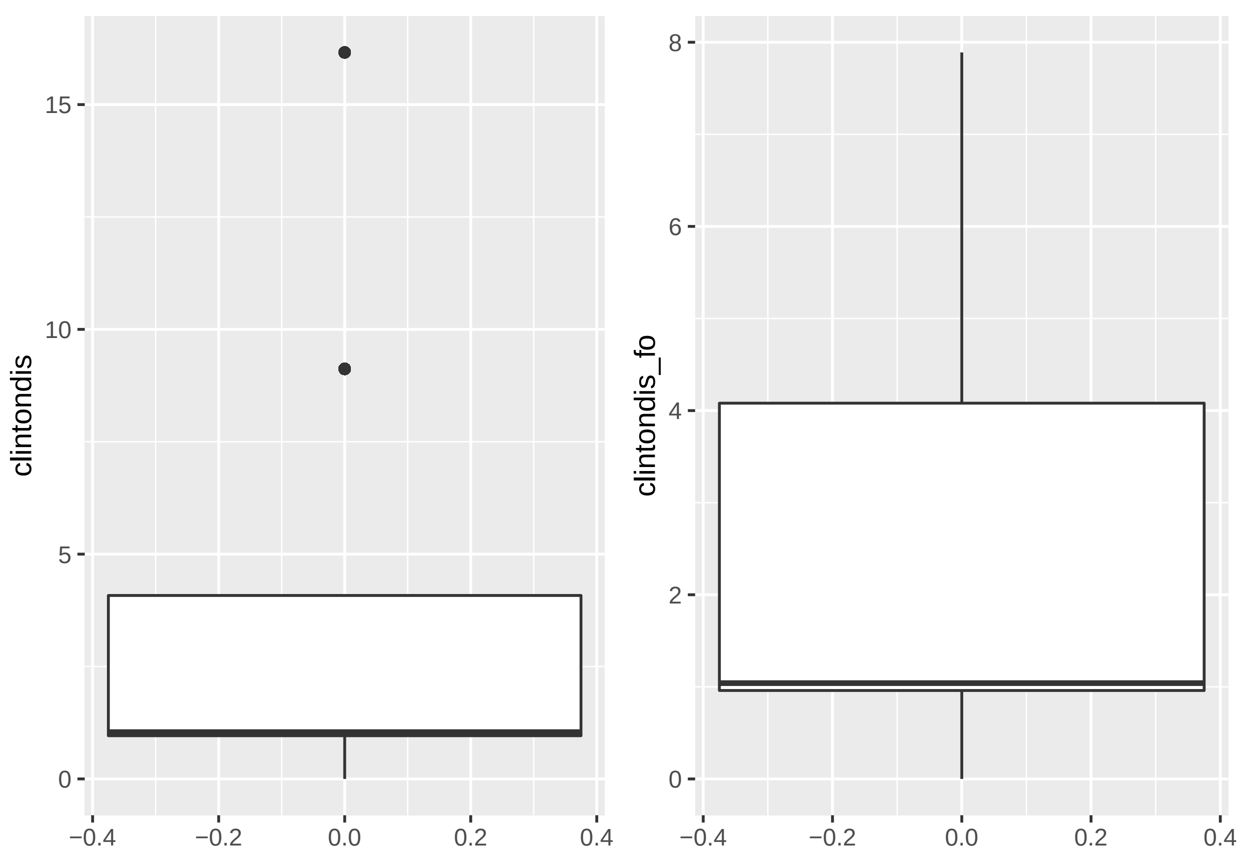
```
vote$natlecon[vote$natlecon==1]<-0
vote$natlecon[vote$natlecon==-1]<-1
vote$natlecon=droplevels(vote$natlecon)
table(vote$natlecon)
```

```
##
##   0   1
## 253 656
```

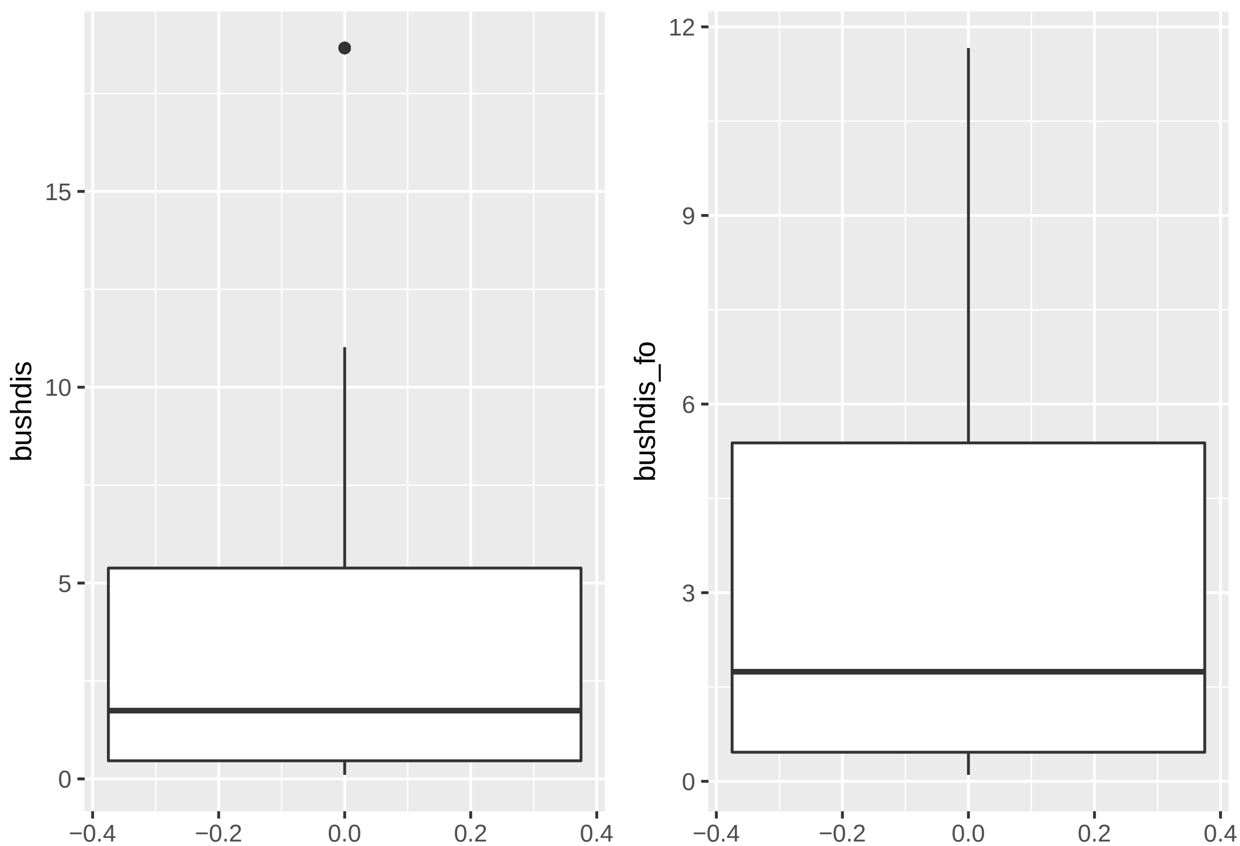
We leave everything as is except for natlecon which has a sparse class regarding the level 1. As solution we combine 0 and 1 as the level 0, meaning national economic conditions have gotten better or stayed the same over the last 12 months. Level -1 gets changed to 1 as well which now means that conditions have gotten better. The change from -1 to 1 is executed just because it is more common to have levels 0 and 1 instead of 0 and -1.

- take care of outliers, treat the skewed distributions and create new features

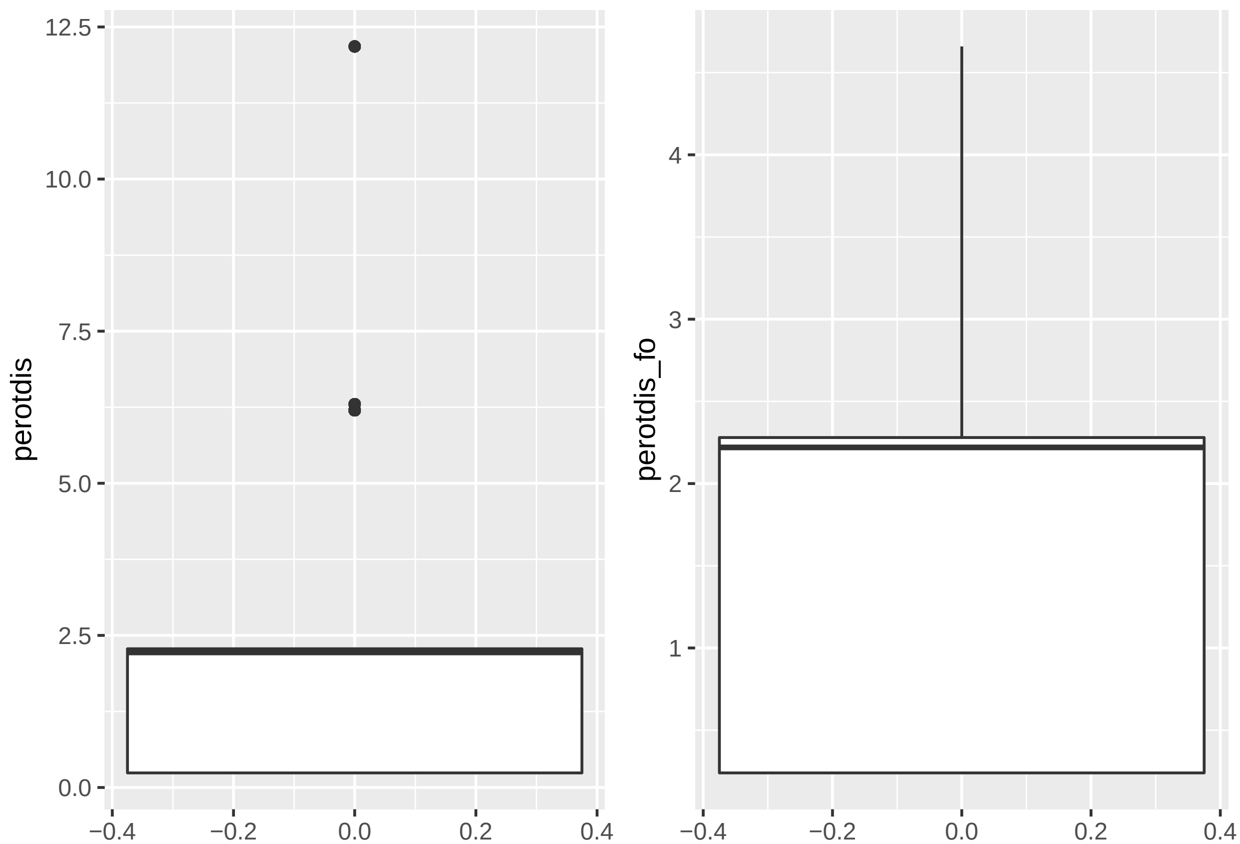
```r
zScores<-function(var) {
    mu<-mean(var)
    sd<-sd(var)
    return((var-mu)/sd)
}

# treating clintondis
tp1<-ggplot(vote,aes(clintondis))+geom_boxplot()+coord_flip()
vote$clintondis_fo<-vote$clintondis
vote$clintondis_fo[zScores(vote$clintondis_fo)>1]<-
    round(mean(vote$clintondis_fo))+sd(vote$clintondis_fo)
tp2<-ggplot(vote,aes(clintondis_fo))+geom_boxplot()+coord_flip()
plot_grid(tp1,tp2,ncol=2)
```



```
# treating bushdis
tp1<-ggplot(vote,aes(bushdis))+geom_boxplot()+coord_flip()
vote$bushdis_fo<-vote$bushdis
vote$bushdis_fo[zScores(vote$bushdis_fo)>2]<-
    round(mean(vote$bushdis_fo))+2*sd(vote$bushdis_fo)
tp2<-ggplot(vote,aes(bushdis_fo))+geom_boxplot()+coord_flip()
plot_grid(tp1,tp2,ncol=2)
```




```
# treating perotdis
tp1<-ggplot(vote,aes(perotdis))+geom_boxplot()+coord_flip()
vote$perotdis_fo<-vote$perotdis
vote$perotdis_fo[zScores(vote$perotdis_fo)>1]<-
    round(mean(vote$perotdis_fo))+sd(vote$perotdis_fo)
tp2<-ggplot(vote,aes(perotdis_fo))+geom_boxplot()+coord_flip()
plot_grid(tp1,tp2,ncol=2)
```

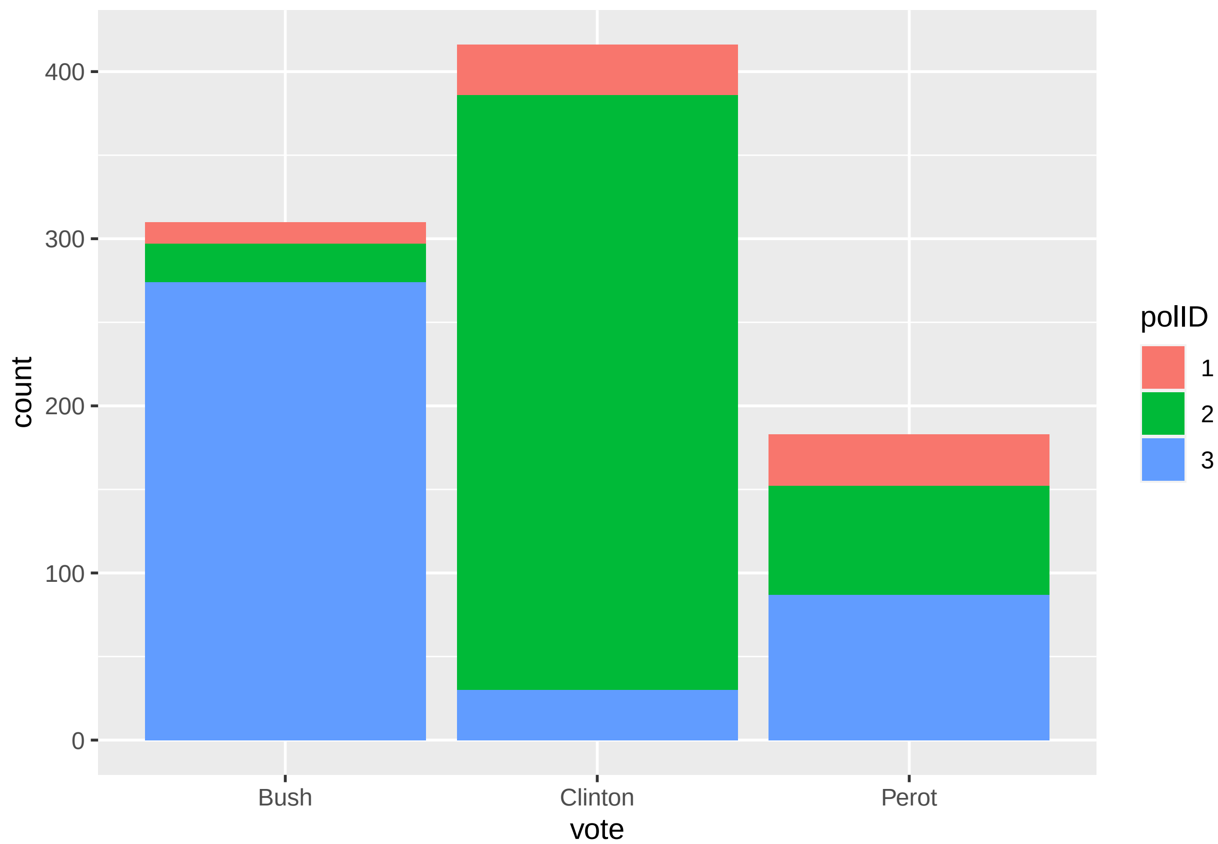



```
# !!make this shit beautiful
# !!reduce the outlier fixes with function
```

There are a few outliers in the variables clintondis, bushdis and perotdis. We fix those outliers and save the fixed data in the variables called [original_var_name]_fo. The ending "fo" is derived from "fixed outliers".

- explore the relationships between predictors and the target

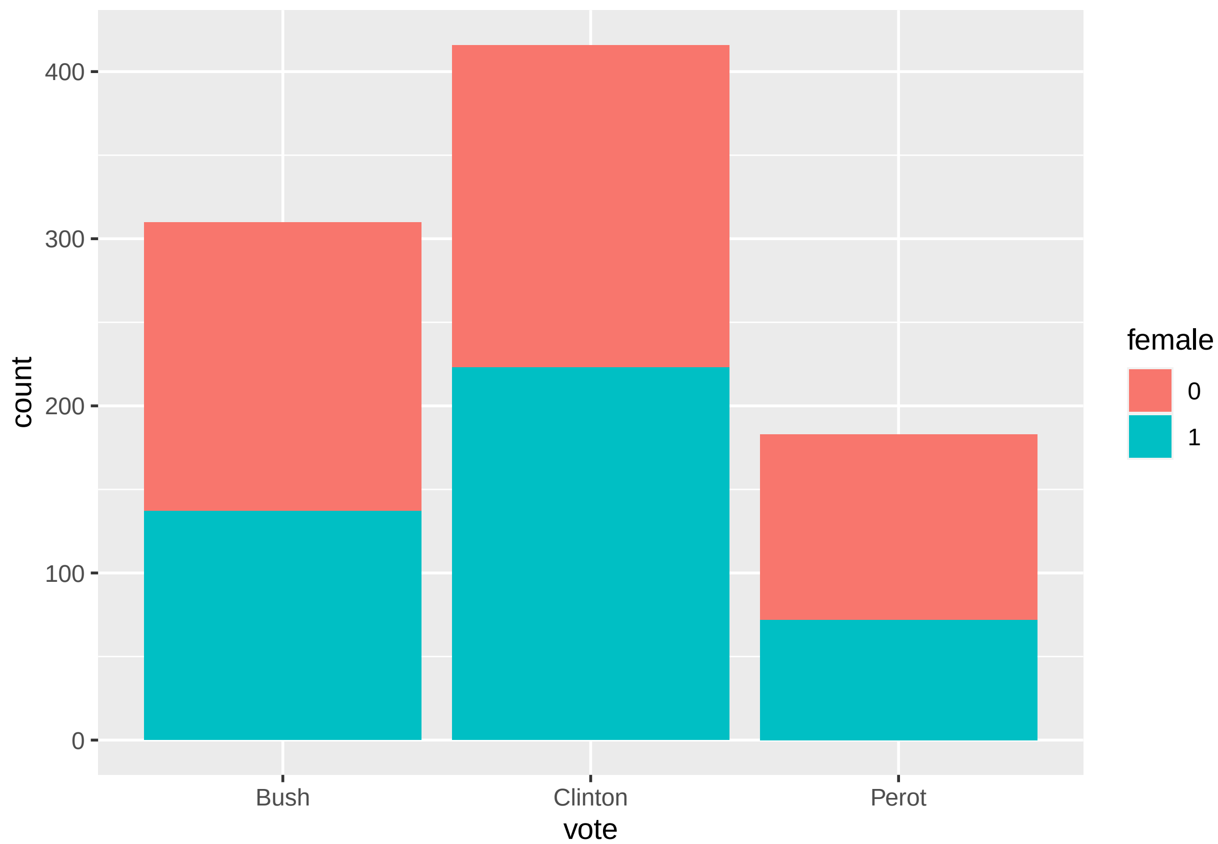
```
ggplot(vote,aes(vote,fill=polID))+geom_bar() # !!fix colours and descriptions
```




```
# !!add percentiles to those splitted barplots somehow
```

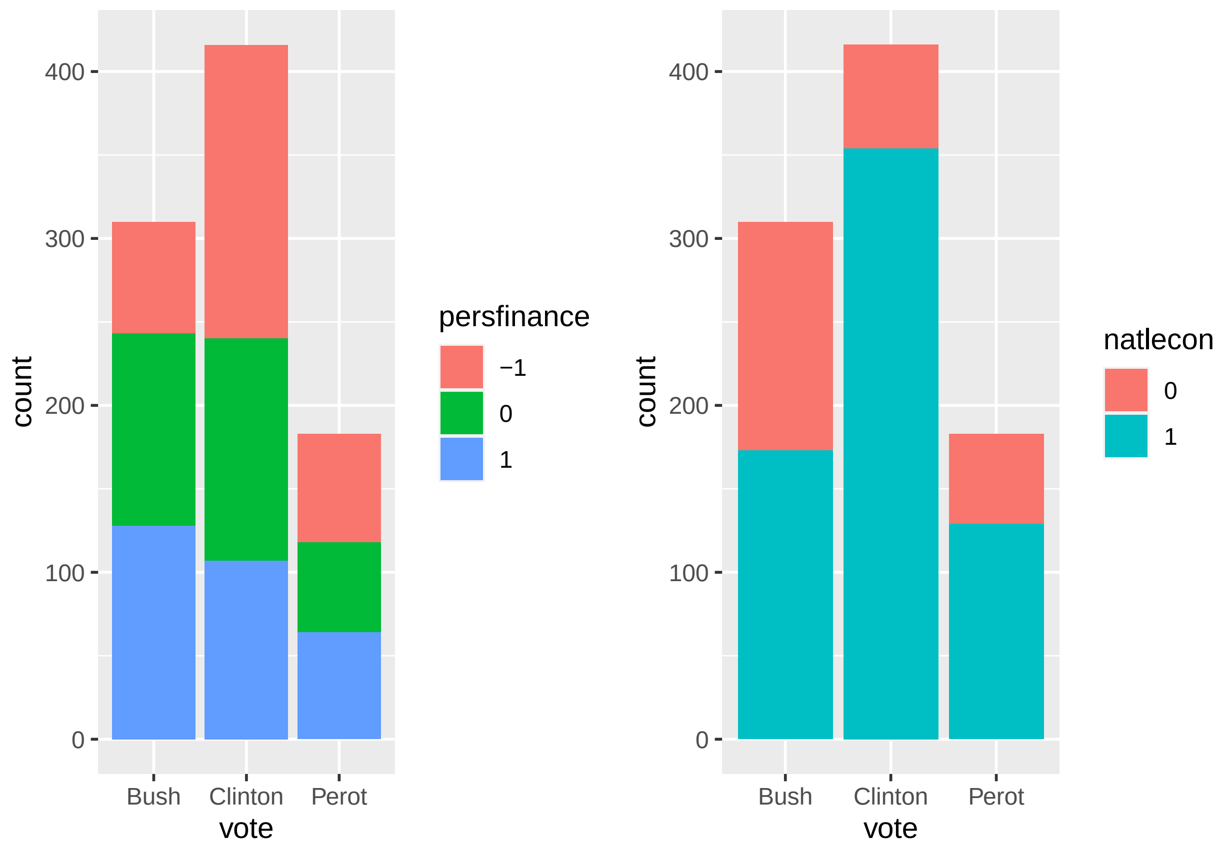
INSERT

```
ggplot(vote,aes(vote,fill=female))+geom_bar()
```



INSERT

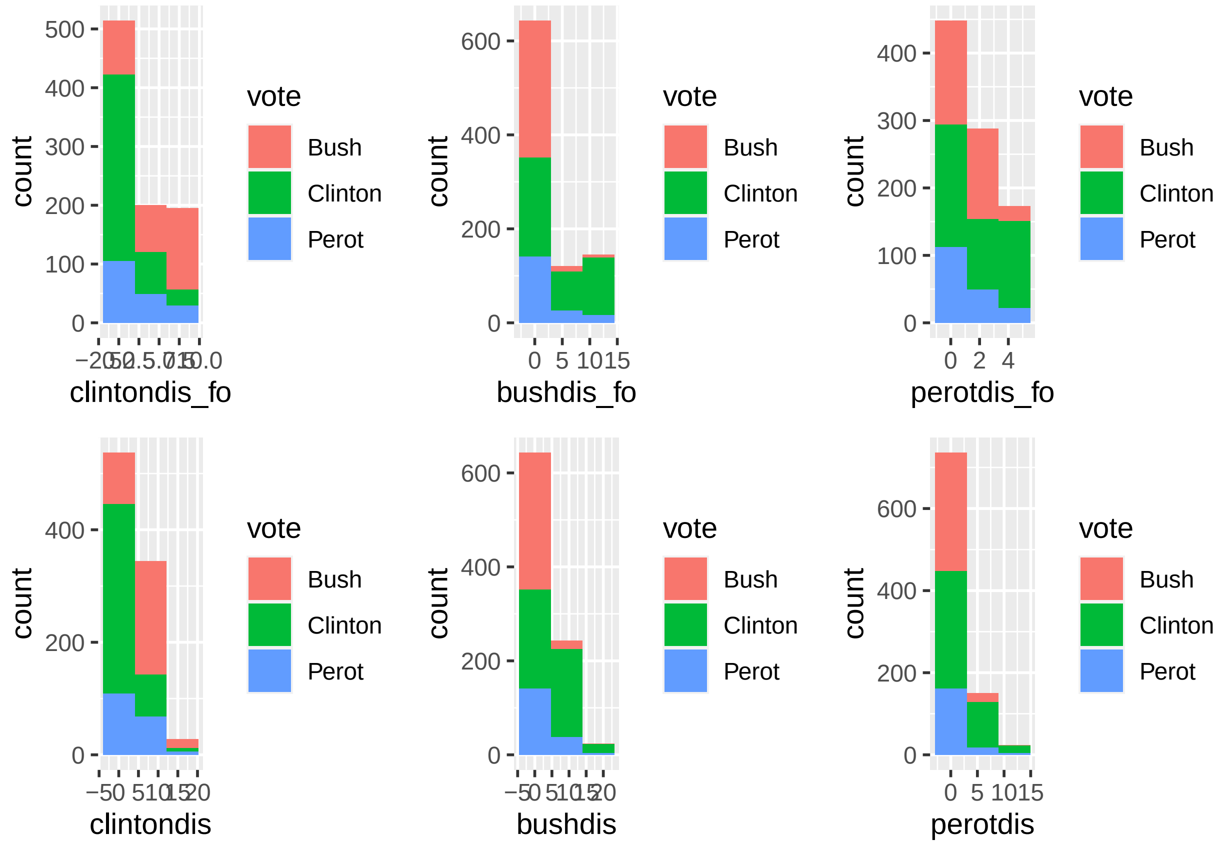
```
p1<-ggplot(vote,aes(vote,fill=persfinance))+geom_bar()
p2<-ggplot(vote,aes(vote,fill=natlecon))+geom_bar()
plot_grid(p1,p2,ncol=2)
```




```
# !!fix repeating barplot by creating a more diverse visual representation
```

INSERT

```
p1<-ggplot(vote,aes(clintondis_fo,fill=vote))+geom_histogram(bins=3)
p2<-ggplot(vote,aes(bushdis_fo,fill=vote))+geom_histogram(bins=3)
p3<-ggplot(vote,aes(perotdis_fo,fill=vote))+geom_histogram(bins=3)
p4<-ggplot(vote,aes(clintondis,fill=vote))+geom_histogram(bins=3)
p5<-ggplot(vote,aes(bushdis,fill=vote))+geom_histogram(bins=3)
p6<-ggplot(vote,aes(perotdis,fill=vote))+geom_histogram(bins=3)
plot_grid(p1,p2,p3,p4,p5,p6,ncol=3)
```




```
# !!fix legend print and axis descriptions, also colours
```

INSERT

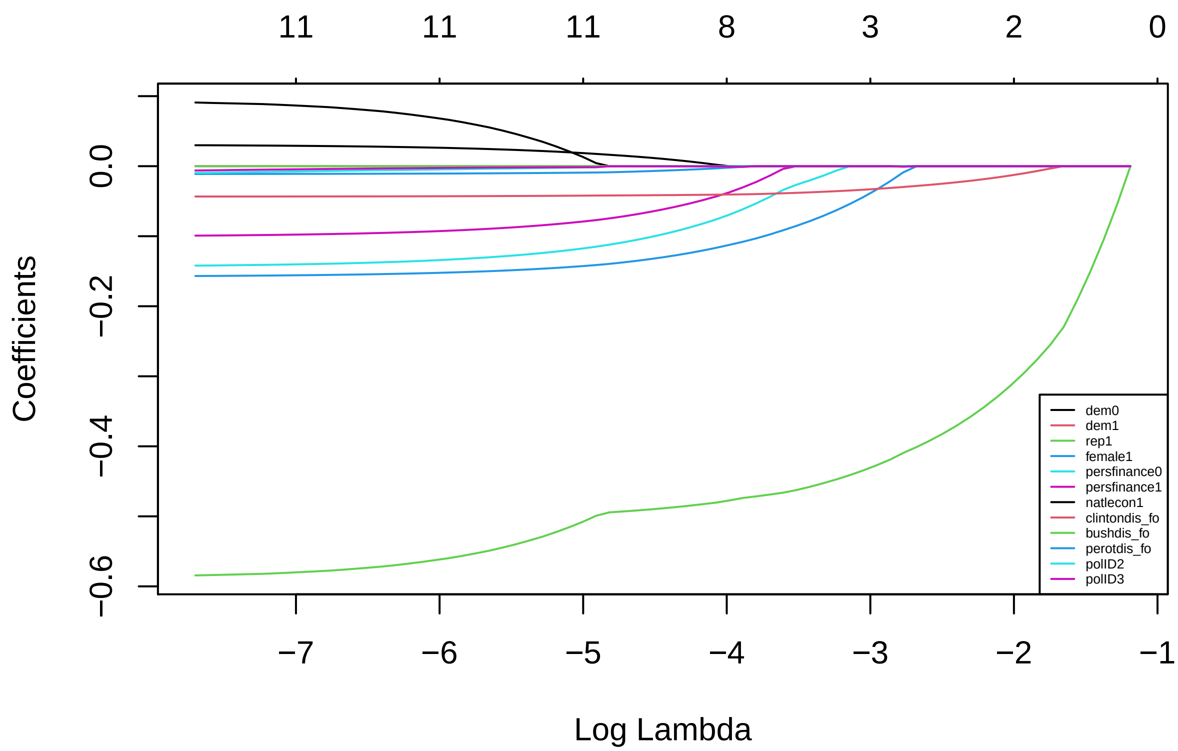
Building the models

```
# splitting the data into train and test
set.seed(777)
train.Index <-  sample(1:nrow(vote), round(0.7*nrow(vote)), replace = F)
# creating the train and test sets using train.Index
vote.train <- vote[train.Index,]
vote.test  <- vote[-train.Index,]

# creating x and y for model training
# y - a target vector
y.train <- vote.train$vote_num
y.test  <- vote.test$vote_num

# X - a matrix with features/predictors
features <- c('dem','rep','female','persfinance','natlecon',
              'clintondis_fo', 'bushdis_fo','perotdis_fo','polID')
```

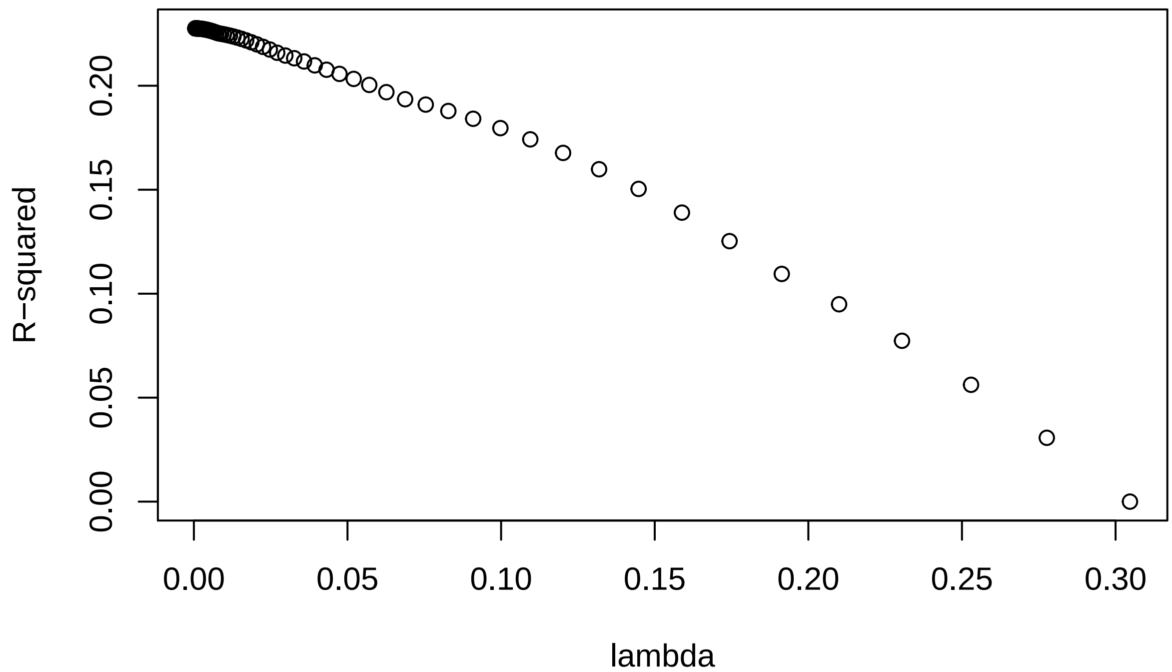
```r
#model.matrix( ~ ., data = scoring.train[, features])
X.train <- model.matrix( ~ . -1, data = vote.train[, features])
X.test  <- model.matrix( ~ . -1, data = vote.test[, features])
```

1. L1-nrom / Lasso

```r
log_l1 <- glmnet(X.train, y.train, alpha = 1)

plot(log_l1, xvar = "lambda")
legend("bottomright", lwd = 1, col = 1:6, legend = colnames(X.train), cex = .4)
```




```r
plot(y = log_l1$dev.ratio,
     x = log_l1$lambda,
     xlab = "lambda",
     ylab = "R-squared")
```

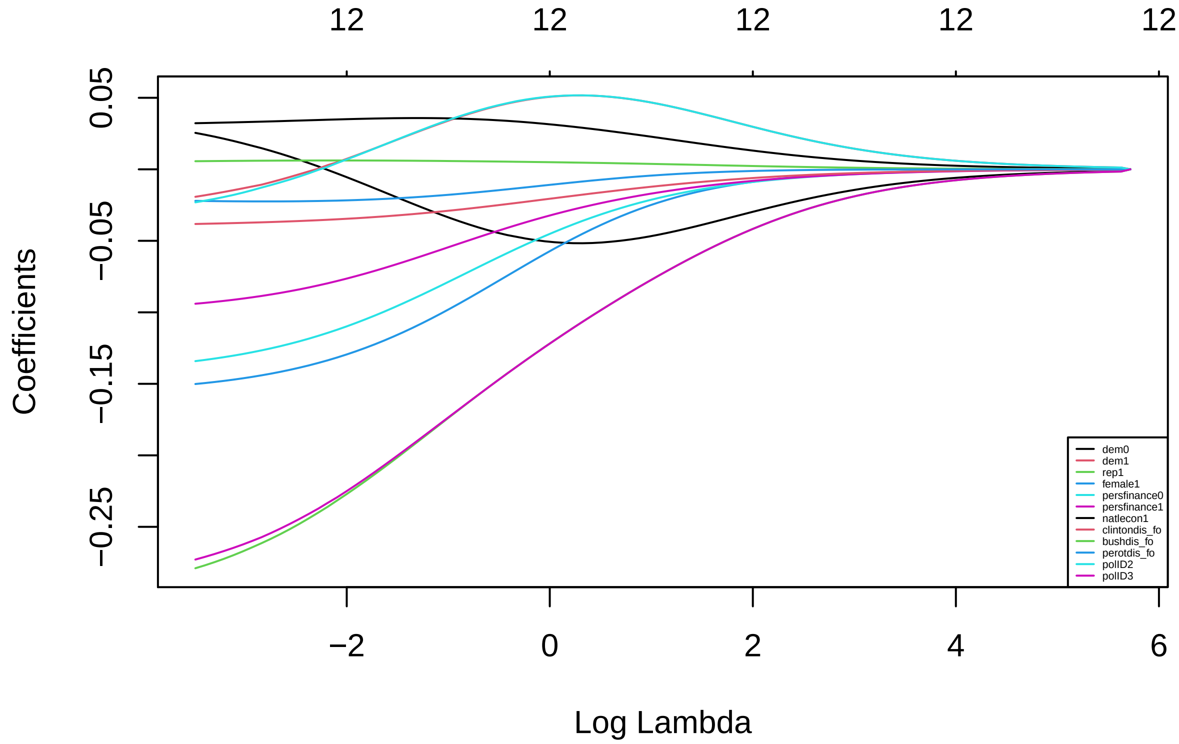



```r
# selecting the optimal lambda
set.seed(77)
log_l1_cv <- cv.glmnet(X.train, y.train, alpha = 1, type.measure = "class",
                      lambda = 10^seq(-5, 1, length.out = 100) , nfolds = 10)
```

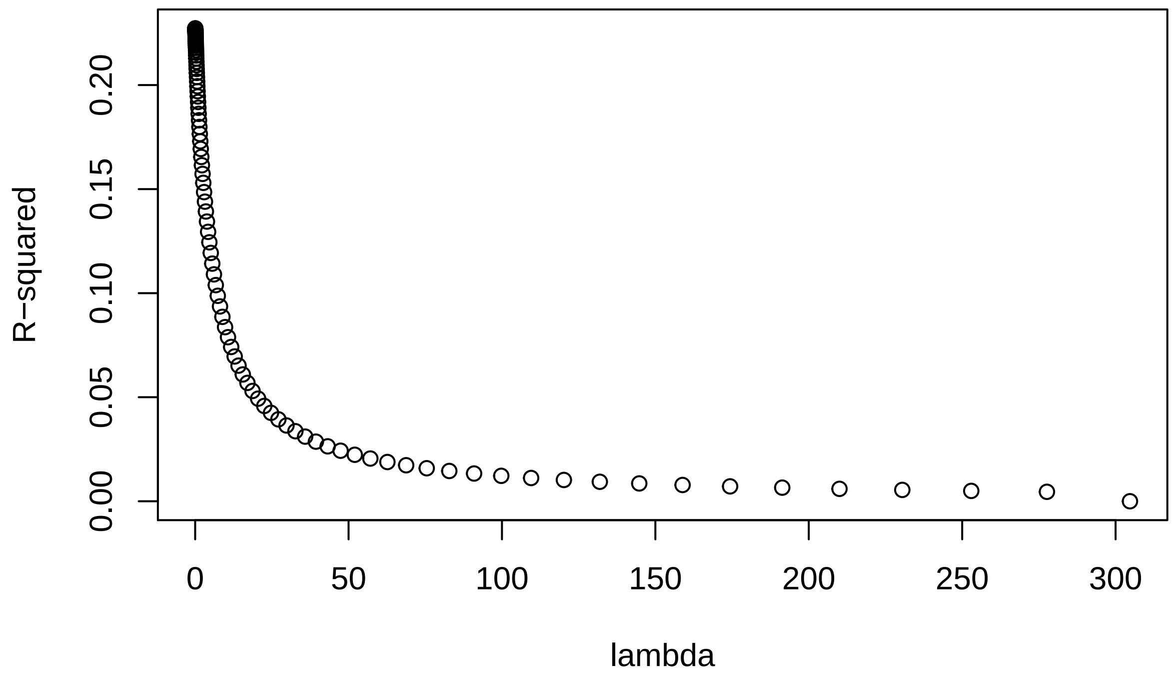
```
## Warning: Only mse, deviance, mae available as type.measure for Gaussian models;
## mse used instead
```

```r
y.predlog_l1 <-  predict(log_l1, newx = X.test,
                        type = "response", s = log_l1_cv$lambda.min)
```

```r
# Setting alpha = 0 implements ridge regression
log_r1 <- glmnet(X.train, y.train, alpha = 0)


plot(log_r1, xvar = "lambda")
legend("bottomright", lwd = 1, col = 1:6, legend = colnames(X.test), cex = .3)
```




```
plot(y = log_r1$dev.ratio,
     x = log_r1$lambda,
     xlab = "lambda",
     ylab = "R-squared")
```




```
# selecting the optimal lambda
set.seed(77)
log_r1_cv <- cv.glmnet(X.train, y.train, alpha = 0, type.measure = "class",
                       lambda = 10^seq(-5, 1, length.out = 100),
                        nfolds = 10)
```

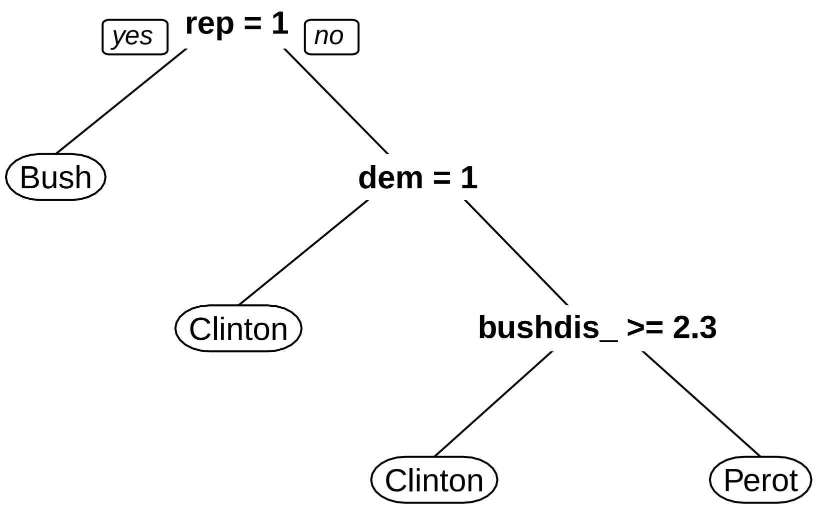
```
## Warning: Only mse, deviance, mae available as type.measure for Gaussian models;
## mse used instead
```

```
y.predlog_r1 <-  predict(log_r1,    newx = X.test,
                         type = "response", s = log_r1_cv$lambda.min)
```

```
#only both

log1 <- glm(vote_num ~  dem + rep + female + persfinance +
              natlecon + clintondis_fo + persfinance + perotdis_fo +bushdis_fo,
           data = vote)
#catagorical
log2 <- glm(vote_num ~ dem + rep + female +persfinance +
          natlecon ,
           data = vote)

#only continous
log3 <- glm(vote_num ~ clintondis_fo + bushdis_fo +
              perotdis_fo,
```

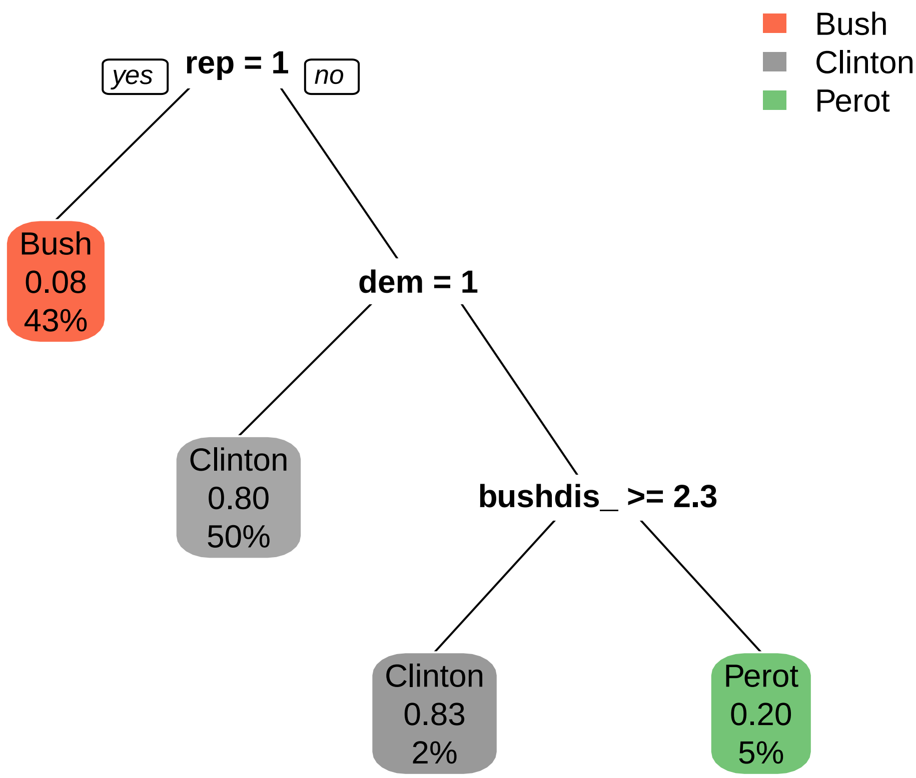
```r
           data = vote)


pred.log1 <- predict(log1, vote, type = "response")
pred.log2 <- predict(log2, vote, type = "response")
pred.log3 <- predict(log3, vote, type = "response")
```

```r
set.seed(7)
train.Index <- caret::createDataPartition(vote$vote, p = 0.7, list = F)
vote.train <- vote[ train.Index,]
vote.test  <- vote[-train.Index,]

# features to be used for model training
features <- c('vote', 'dem','rep','female','persfinance','natlecon',
              'clintondis_fo', 'bushdis_fo','perotdis_fo','polID')

# ----- Fitting a model ------
# Training classification decision tree
dt <- rpart(vote ~ .,
            data = vote.train[,features],
            method = "class",   #cause we have a classification problem
            parms = list(split = "information"),  # the splitting index
            model = T)

# ----- Deriving Predictions ------

pred.dt <- predict(dt, newdata = vote.test, type = "prob")[, 2]

# Visualizing the results from "dt" using the prp() function
# default plot
prp(dt)
```




```
# prints the percentage of observations and class probabilities in each node
prp(dt, extra = 106, border.col = 0, box.palette="auto")
```

```
## Warning: extra=106 but the response has 3 levels (only the 2nd level is
## displayed)
```



We decided to create a few models for the data set vote. We trained a portion of the data set to create a lasso and ridge model. We also created 3 logistic regression models. 2 that only have either categorical or continues variables and the other one that has a combination of both. We also trained a nother portion of the data set for our decision tree and we created it.

Predictions

```
Accuracy <- function(pred, real, threshold = 0.5){
  predClass <-  ifelse(pred > threshold, 1, 0)
  acc <- sum(predClass == real) / length(real)
  return(acc)
}
```

```
(acc1 <- Accuracy(pred = pred.log1, real = vote$vote_num))
```

```
## [1] 0.3410341
```

```
(acc2 <- Accuracy(pred = pred.log2, real = vote$vote_num))
```

```
## [1] 0.3410341
```

```r
(acc3 <- Accuracy(pred = pred.log3, real = vote$vote_num))
```

```
## [1] 0.3410341
```

```r
# Brier Score
(BS.log1 <- sqrt(mean((vote$vote_num - pred.log1)^2)))
```

```
## [1] 0.6459124
```

```r
(BS.log2 <- sqrt(mean((vote$vote_num- pred.log2)^2)))
```

```
## [1] 0.6552625
```

```r
(BS.log3 <- sqrt(mean((vote$vote_num - pred.log3)^2)))
```

```
## [1] 0.6835589
```

As we can see here the accuracy score for all the logistic regression is the same. Since that is the case we need to focus on the brier score. Which would mean that log1 which is the one that has both continouse and catagorical variables in it is the best one of the bunch.

```r
(accLasso <- Accuracy(pred = y.predlog_l1, real = y.test))
```

```
## [1] 0.3516484
```

```r
(accLRidge <- Accuracy(pred = y.predlog_r1, real = y.test))
```

```
## [1] 0.3516484
```

```r
(BS.logL1 <- sqrt(mean((y.test - y.predlog_l1)^2)))
```

```
## [1] 0.6852876
```

```r
(BS.logL2 <- sqrt(mean((y.test - y.predlog_r1)^2)))
```

```
## [1] 0.6852333
```

With the Lasso and ridge model it is the same case the accuracy scores r identical and since that is the case we need to choose the model with the lowest brier score which would be the ridge model.

```r
# ----- Evaluating Prediction Quality -----
# Calculate performance with AUC and RMSE
auc(vote.test$vote_num, pred.dt)
```

```
## Warning in roc.default(response, predictor, auc = TRUE, ...): 'response'
## has more than two levels. Consider setting 'levels' explicitly or using
## 'multiclass.roc' instead
```

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```

```
## Area under the curve: 0.9299
```

```
( rmse <- sqrt(mean((vote.test$vote_num - pred.dt)^2)) )
```

```
## [1] 1.565776
```

```
Accuracy(pred=pred.dt, real=vote.test$vote_num)
```

```
## [1] 0.01845018
```

```
# Naive Classifier
baseline_probability <- sum(vote.train$vote_num == 1)/nrow(vote.train)
pred.baseline <- rep(baseline_probability, nrow(vote.test))

auc(vote.test$vote_num, pred.baseline)
```

```
## Warning in roc.default(response, predictor, auc = TRUE, ...): 'response'
## has more than two levels. Consider setting 'levels' explicitly or using
## 'multiclass.roc' instead
```

```
## Setting levels: control = 1, case = 2
## Setting direction: controls < cases
```

```
## Area under the curve: 0.5
```

```
( rmse <- sqrt(mean((vote.test$vote_num - pred.baseline)^2)) )
```

```
## [1] 1.679247
```

```
Accuracy(pred=pred.baseline, real=vote.test$vote_num)
```

```
## [1] 0
```

Here tbh i dont know wt to say lovis ;D