What Factors Affect Mental Health Treatment in the Workforce

Smruthi Iyengar

4/30/2019

Introduction

This data set describes the workforce and their relationship to mental health. It contains 993 observations from a survey conducted in August 2014.The point of this survey was to see what factors in a person’s workplace affect a person’s decsion to obtain mental health treatment. The Y variable is treatment and that represents how many people have recieved mental health treatment. The x variables I used were family history,work interference,benefits,care options, gender and anonymity. The goal of this project is to see if workforce characteristics (x variables) can determine whether or not someone has gotten help for their mental health. It should be noted that all these variables are factor variables. I choose these variables because I thought that they would be the most relevant to study.Earlier, I cleaned the dataset and found that these variables seemed to have the most significance in determining whether someone gets mental health help.I choose family history because I thought that if someone has a history of mental health problems then they are more likely to get treatment for mental health. For the gender variable I assumed that women were more likely to get mental health help than men. This is because in society men sometimes might feel like they are weak for getting help. The work interference variable asks if you have a mental health condition does it interfere with your work. For this variable I feel like those who say that it interfers with their work would be more likely to get treatment. The benefits variable asks if their employer provides mental health benefits.I believe that if someone answers yes to this question they are more likely to obtain mental health treatment. The care options variable asks if the person surveyed knows the options for mental health care their employer provides.For this one I believe that those who know their options are more likely to get help. The final x variable i choose was anonymity, this variable asks if the person surveyed anonymity is protected if they choose to take advantage of mental health treatment and resources.

Summary stats and coding the variables.

```
summary(as.numeric(data$treatment))

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   1.000   1.000   2.000   1.635   2.000   2.000
```

The variable Treatment is a factor variable it is coded 1 for No Treatment and 2 for Treatment. The mean is 1.635 so that means 63.5 percent of people interviewed in the survey did recieve mental health treatment.

```
data$family_history <- as.factor(data$family_history)
data$work_interfere <- as.factor(data$work_interfere)
data$benefits <- as.factor(data$benefits)
data$care_options <- as.factor(data$care_options)
data$anonymity <- as.factor(data$anonymity)
data$gender <- as.factor(data$gender)
levels(data$family_history)

## [1] "No"  "Yes"

levels(data$work_interfere)

## [1] "Never"     "Often"     "Rarely"    "Sometimes"

levels(data$benefits)

## [1] "Don't know" "No"         "Yes"

levels(data$care_options)

## [1] "No"       "Not sure" "Yes"

levels(data$anonymity)

## [1] "Don't know" "No"         "Yes"

levels(data$gender)

## [1] "female" "male"   "trans"
```

These are the levels of the other x variables. Family history is a factor variable,it is coded 1 for “No Family History” and 2 for “Family History”

Work inferference is coded 1 for “Never”, 2 for “Often” 3 for “Rarely” and 4 for “Sometimes”

Benefits is coded 1 for “Don’t Know” 2 for “Not sure” and 3 for “Yes”

Care options is coded 1 for “No” 2 for “Not sure” and 3 for “yes”

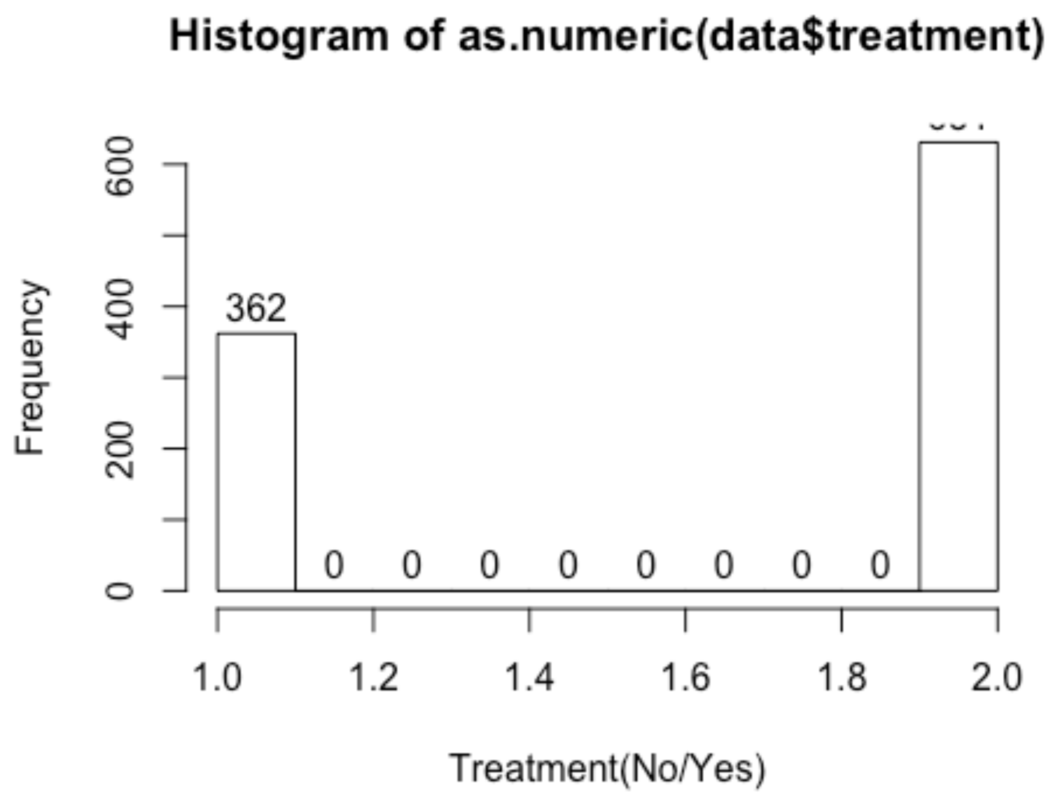
Anonymity is coded 1 for “Don’t know”, 2 for “No” and 3 for “Yes”

Gender is coded 1 for “female” 2 for “male” and 3 for “trans/other”

```
summary(as.numeric(data$family_history))

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    1.00    1.00    1.45    2.00    2.00
```

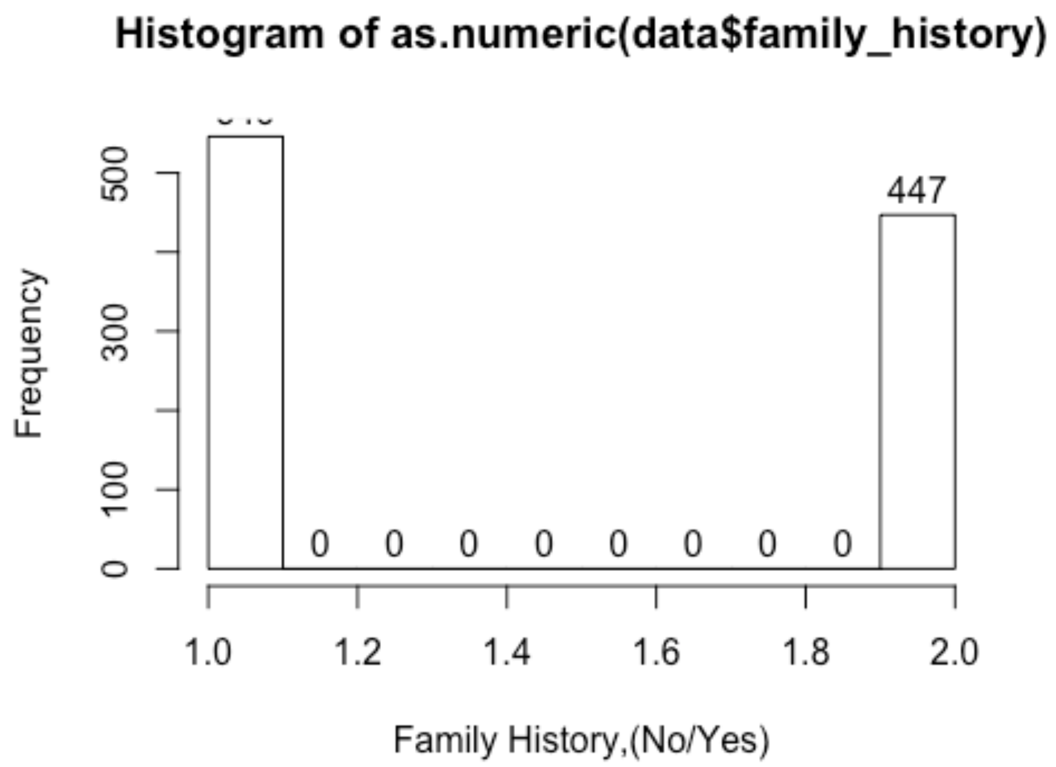
The mean of family history is 1.45, this means that 45% of the people surveyed have a family history of mental health

```r
hist(as.numeric(data$treatment),xlab = "Treatment(No/Yes)",labels = TRUE)
```



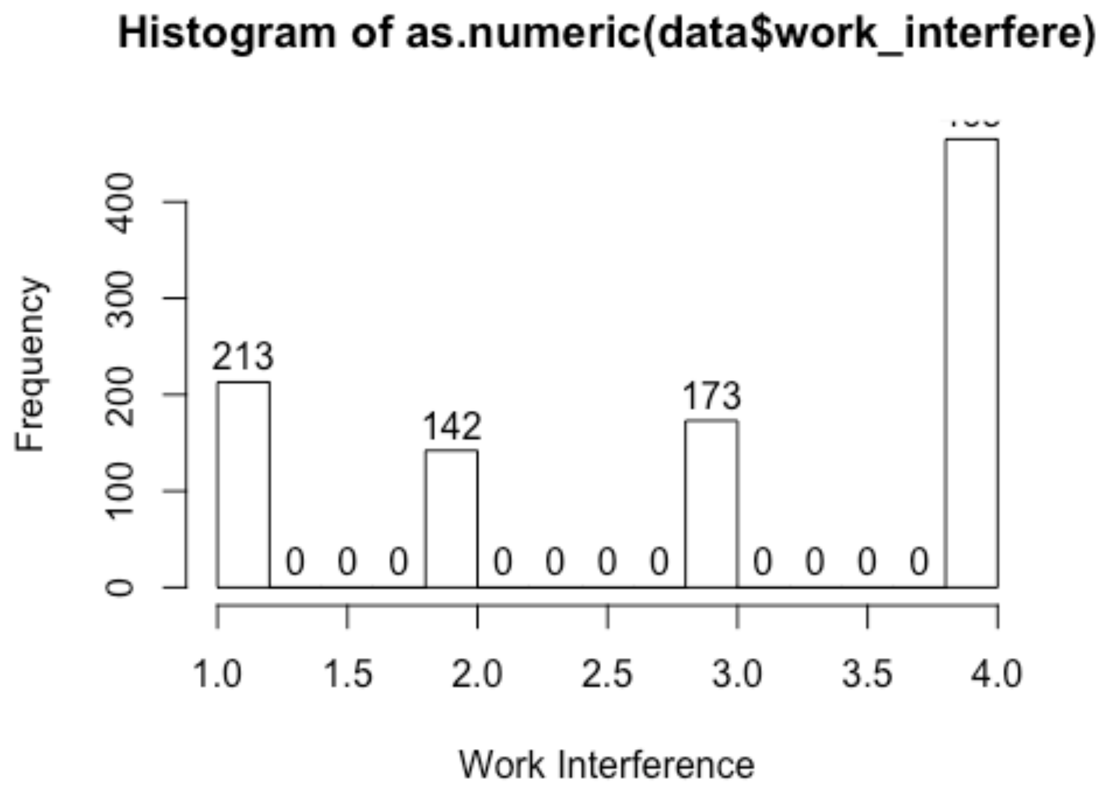
This plot shows that about 36 percent of the the amount of sample did not get treatment and the other part did get treatment.

```r
hist(as.numeric(data$family_history),xlab = "Family History,(No/Yes)",labels
= TRUE)
```



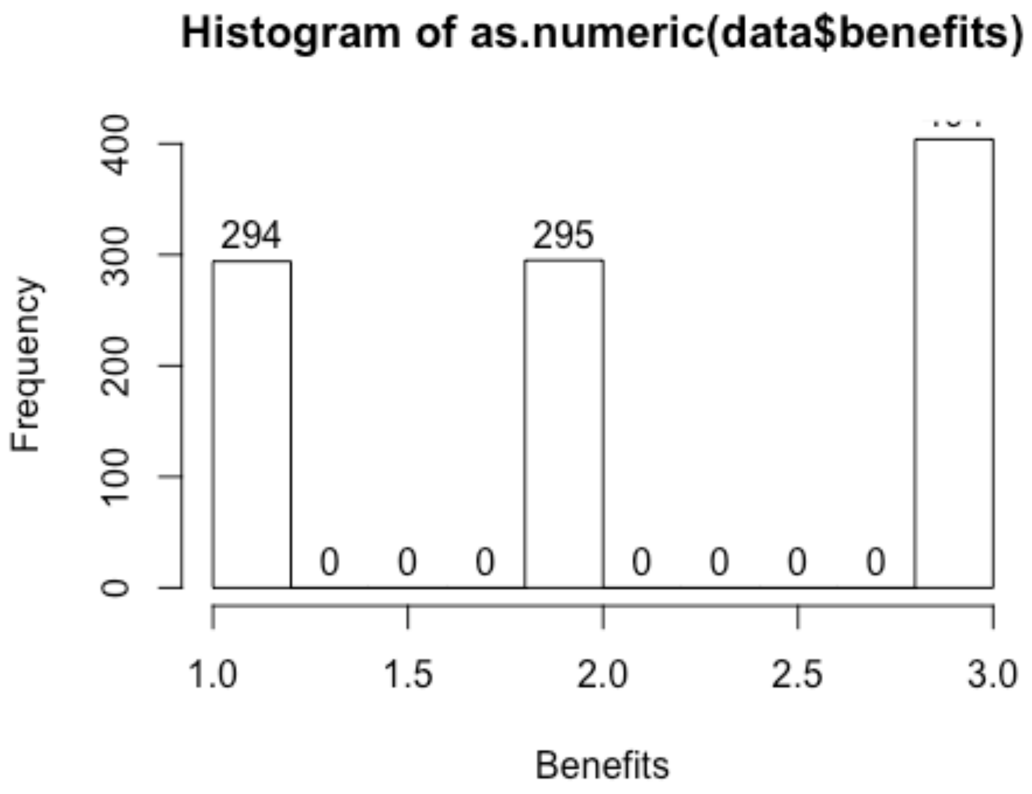
The histogram shows that 447 out of the 993 people surveyed had a family history of mental health problems.

```
hist(as.numeric(data$work_interfere),xlab = "Work Interference",labels = TRUE
)
```



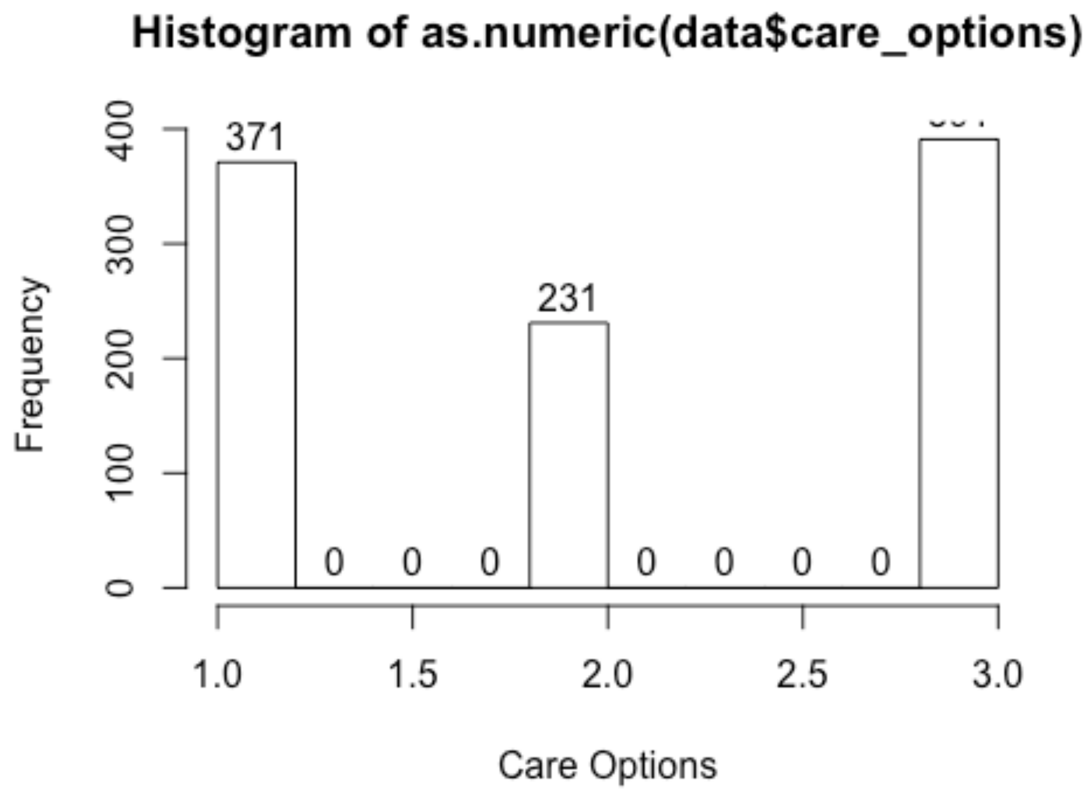
This histogram shows that a majority of people (465) of the 993 people interviewed reported that their mental health sometimes affects their work. The next most popular option was that it Never affected their work. This was followed by it rarely affecting someone work. The least popular choose was it often affecting their work.

```
hist(as.numeric(data$benefits),xlab = "Benefits",labels = TRUE)
```



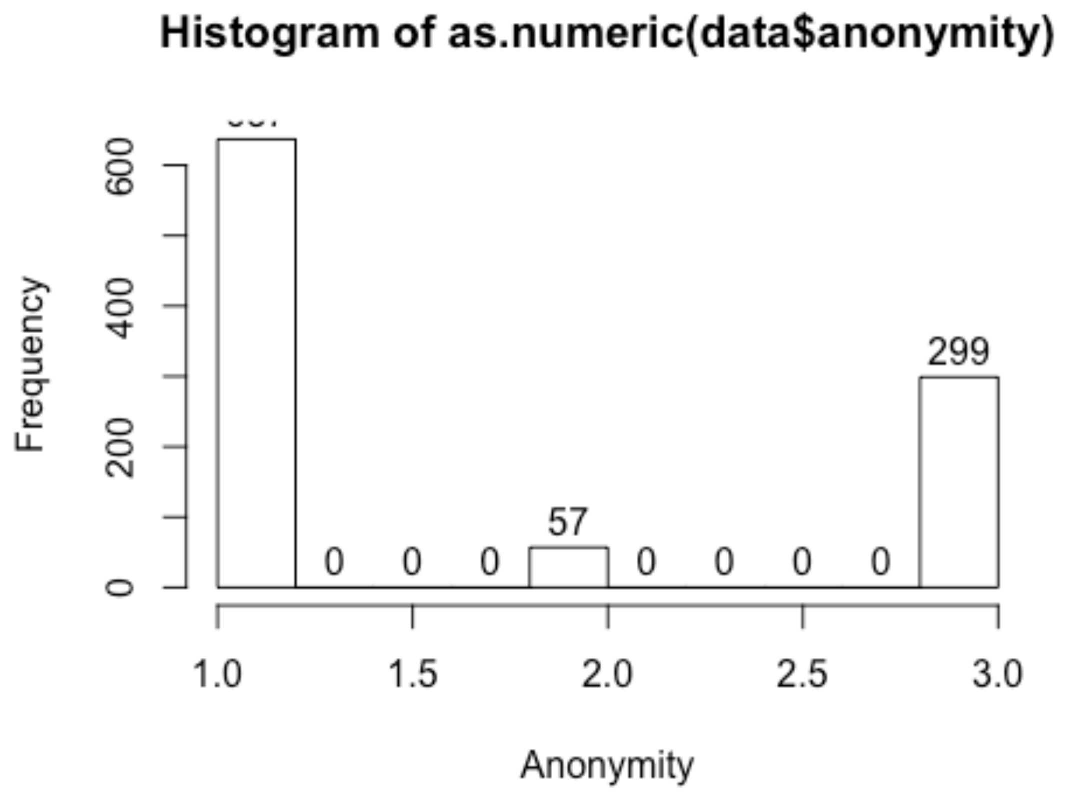
It was pretty evenly divided between 1 (Don't know), 2(No) and 3 (Yes). But 40 percent of people said that their employer does provide benefits.30 percent of people said their employer does not provide benefits and the remaining 30 percent did not know.

```
hist(as.numeric(data$care_options),xlab = "Care Options",labels = TRUE)
```



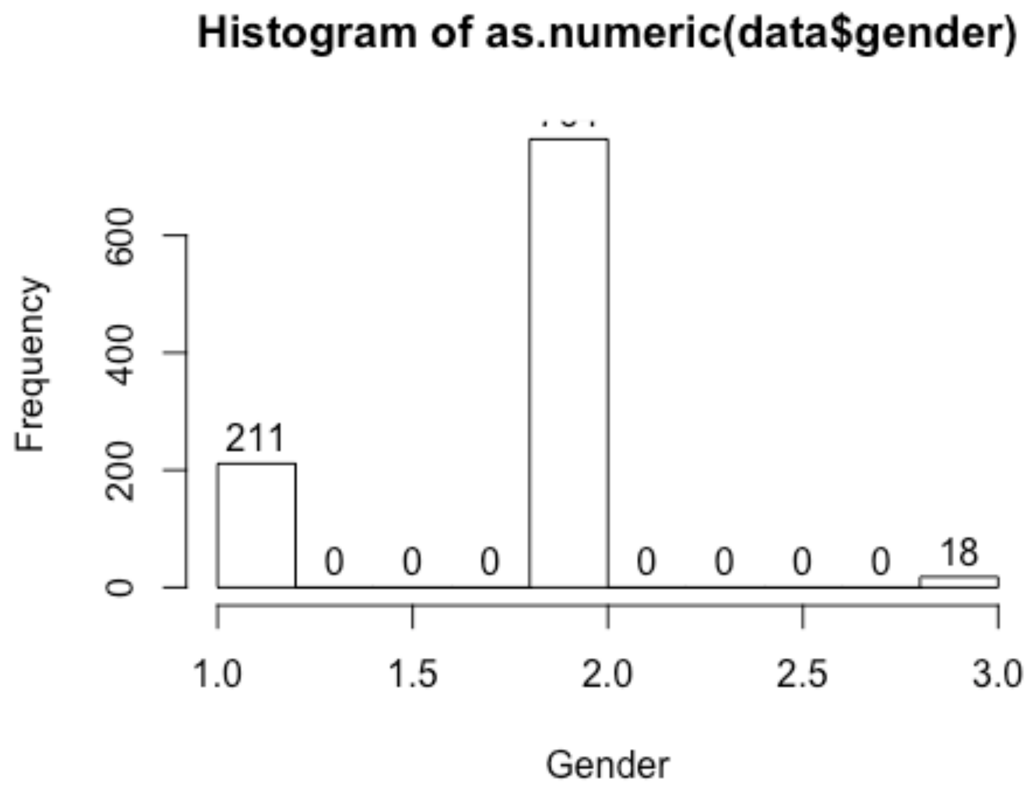
Here 1 represents people who said “No” 2 represents the people who said “not sure” and 3 represents people who said “yes”. About 40 percent of people said that they know the care options they have. 37 percent said that they do not know the care options they have. The remaining 23 percent said they were not sure.

```
hist(as.numeric(data$anonymity),xlab = "Anonymity",labels = TRUE)
```



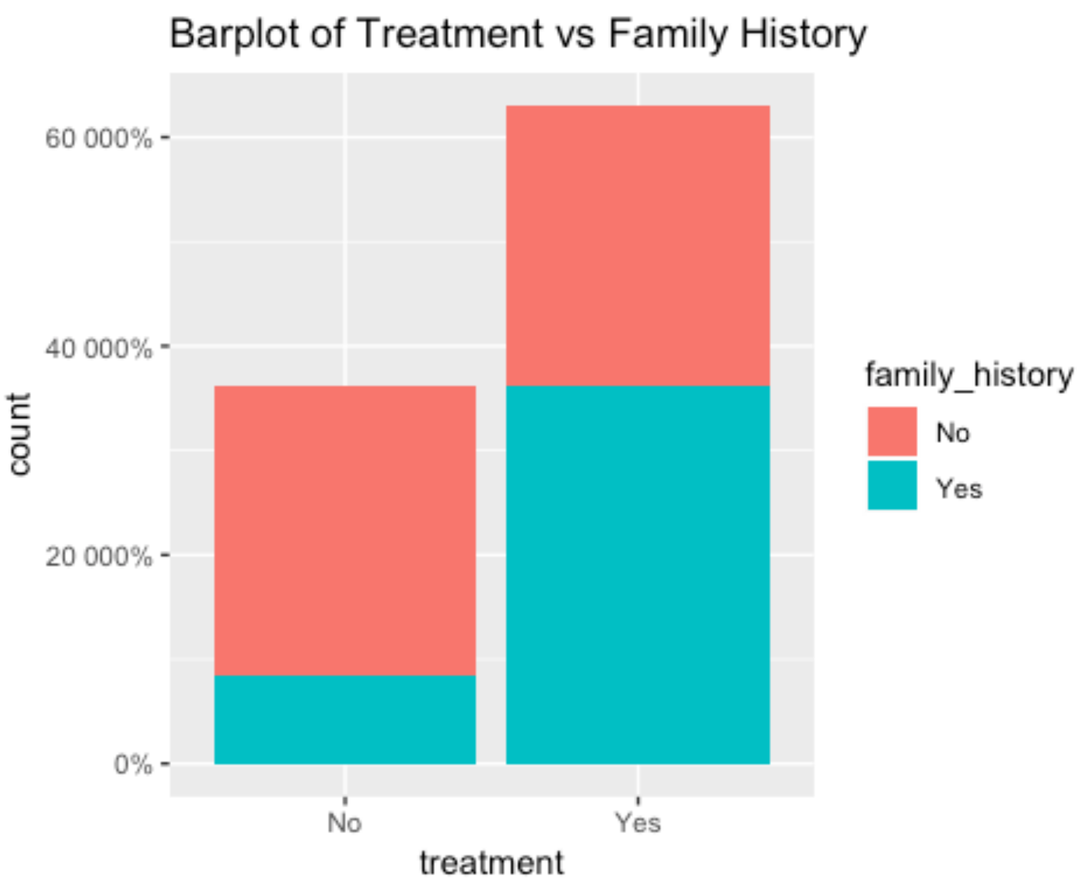
Here 1 stands for “Don’t know” 2 stands for “No” and 3 stands for “Yes”. 64 percent of people did not know if their anonymity was protected if they got mental health treatment. 30 percent of people did know that they would be able to stay anon. The remaining 6 percent said that they know they would not be able to stay anon.

```r
hist(as.numeric(data$gender),xlab = "Gender",labels = TRUE)
```



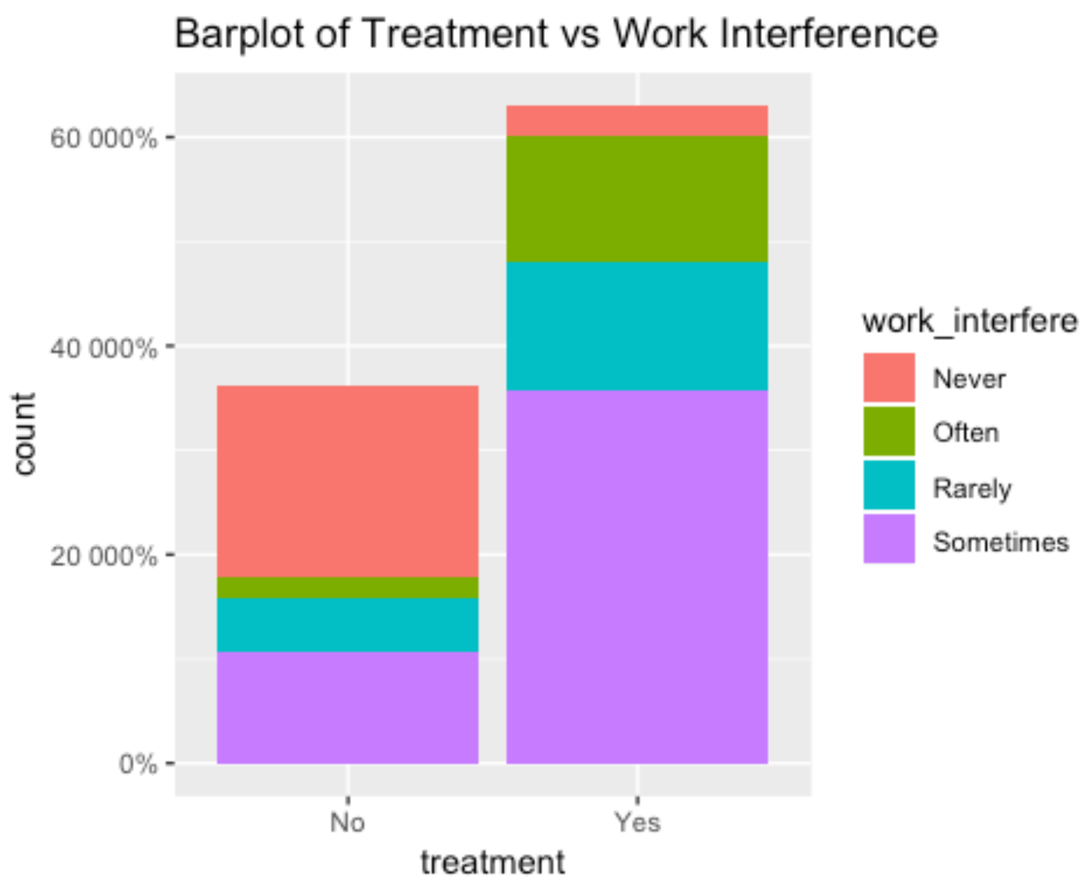
76 Percent of people(754 people) were male and 21 percent of people interviewed were female. 3 percent of people interviewed were trans.

```
#*Plots of treatment(y variable) vs x variables*

library(ggplot2)
library(tidyr)
ggplot(data = data)+geom_bar(aes(x = treatment,fill= family_history))+
  scale_y_continuous(labels = scales::percent)+
  ggtitle(label = "Barplot of Treatment vs Family History")
```



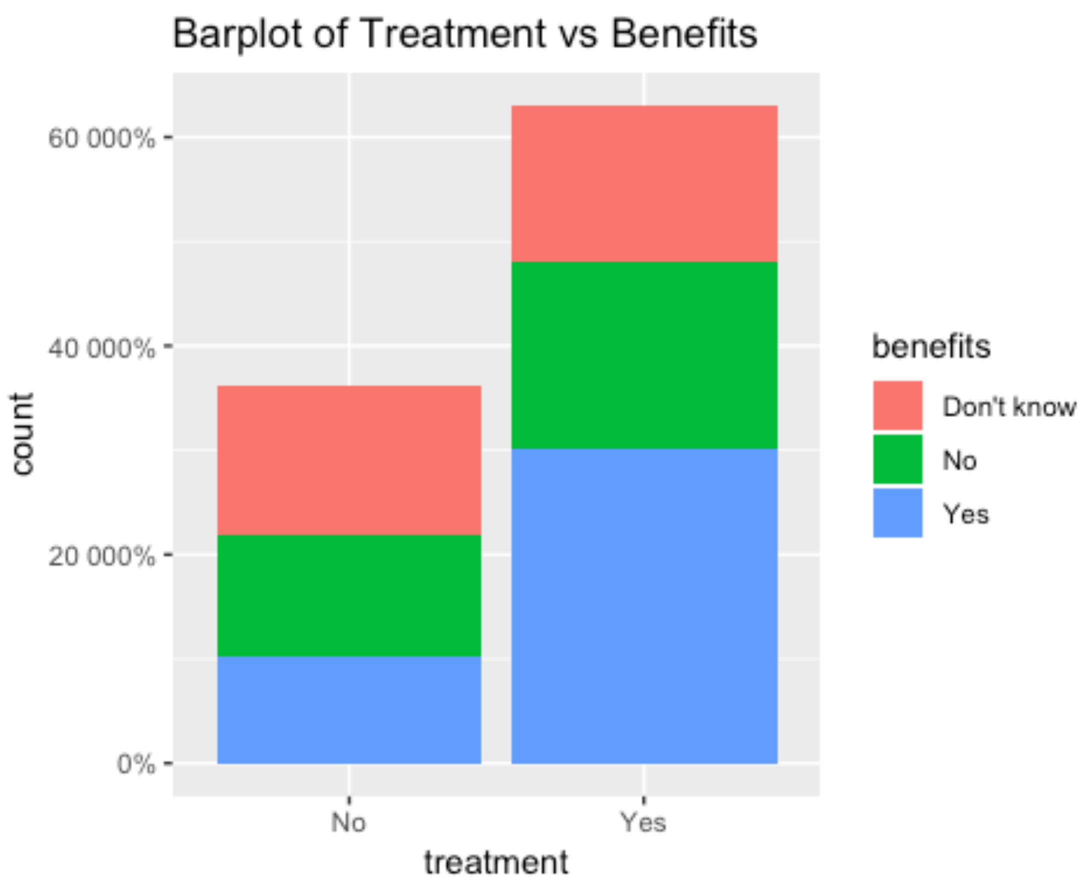
Only about 10 percent of the people surveyed did not get treatment and had a family history of mental health. About 37 percent of the people surveyed did have a family history of mental health and did get treatment. So, those who have a family history are more likely to get treatment

```
ggplot(data = data)+geom_bar(aes(x = treatment,fill= work_interfere))+
  scale_y_continuous(labels = scales::percent)+
  ggtitle(label = "Barplot of Treatment vs Work Interference")
```



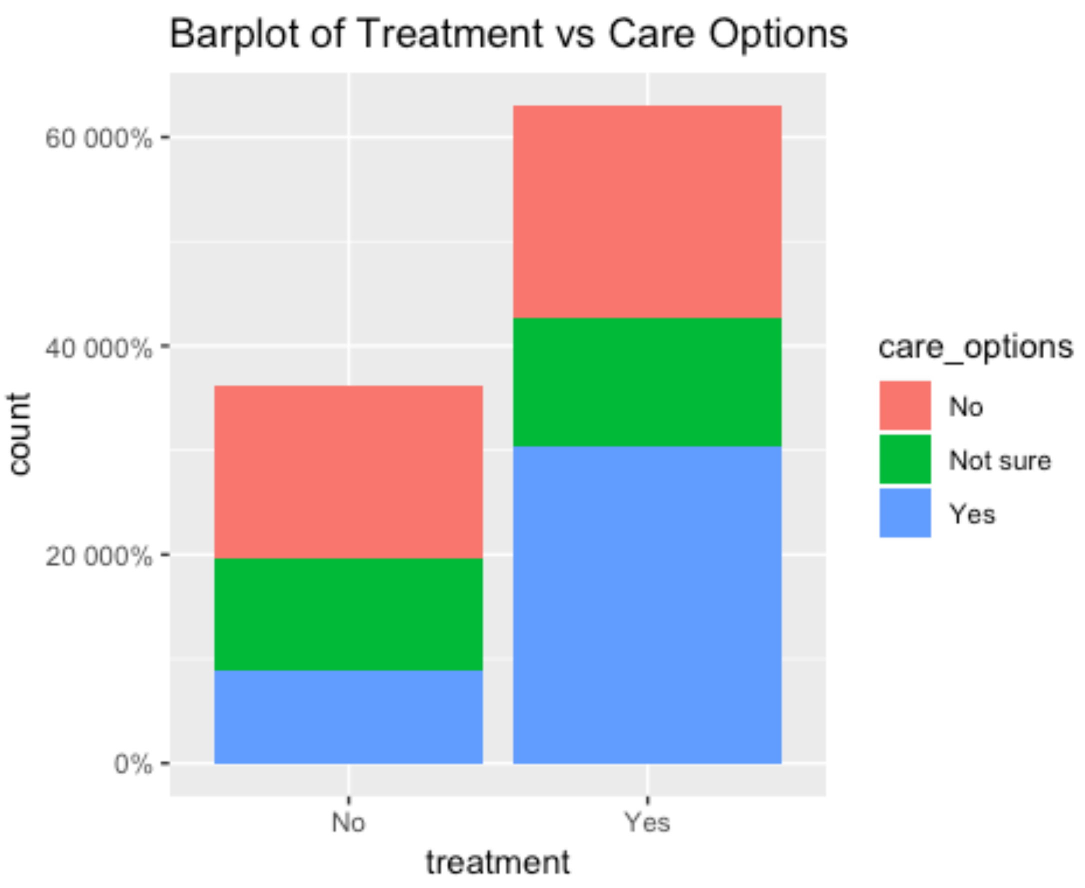
Those who believe that their mental health never affects their work are more likely to not get treatment. But, for all the other factors they are more likely to get mental health treatment.

```
ggplot(data = data)+geom_bar(aes(x = treatment,fill= benefits))+
  scale_y_continuous(labels = scales::percent)+
  ggtitle(label = "Barplot of Treatment vs Benefits")
```



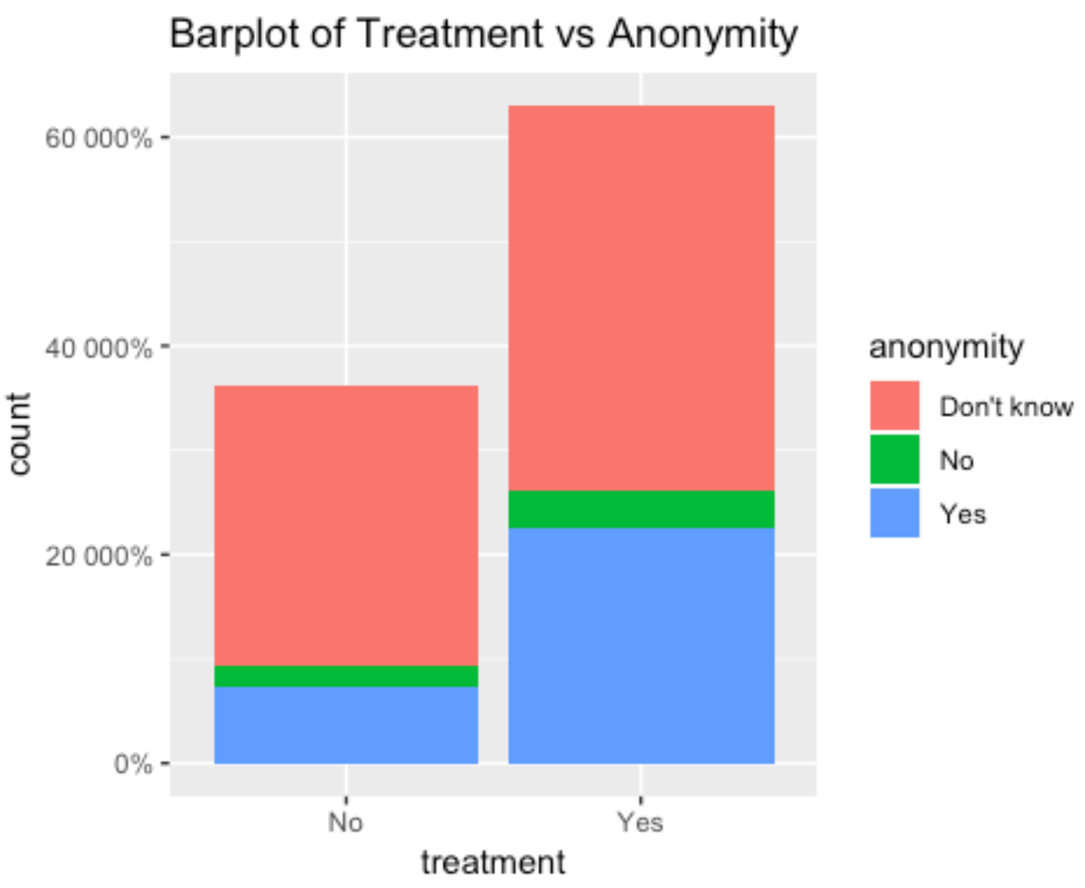
If someone knows that their work has benefits they are more likely to get treatment.

```
ggplot(data = data)+geom_bar(aes(x = treatment,fill= care_options))+
  scale_y_continuous(labels = scales::percent)+
  ggtitle(label = "Barplot of Treatment vs Care Options")
```



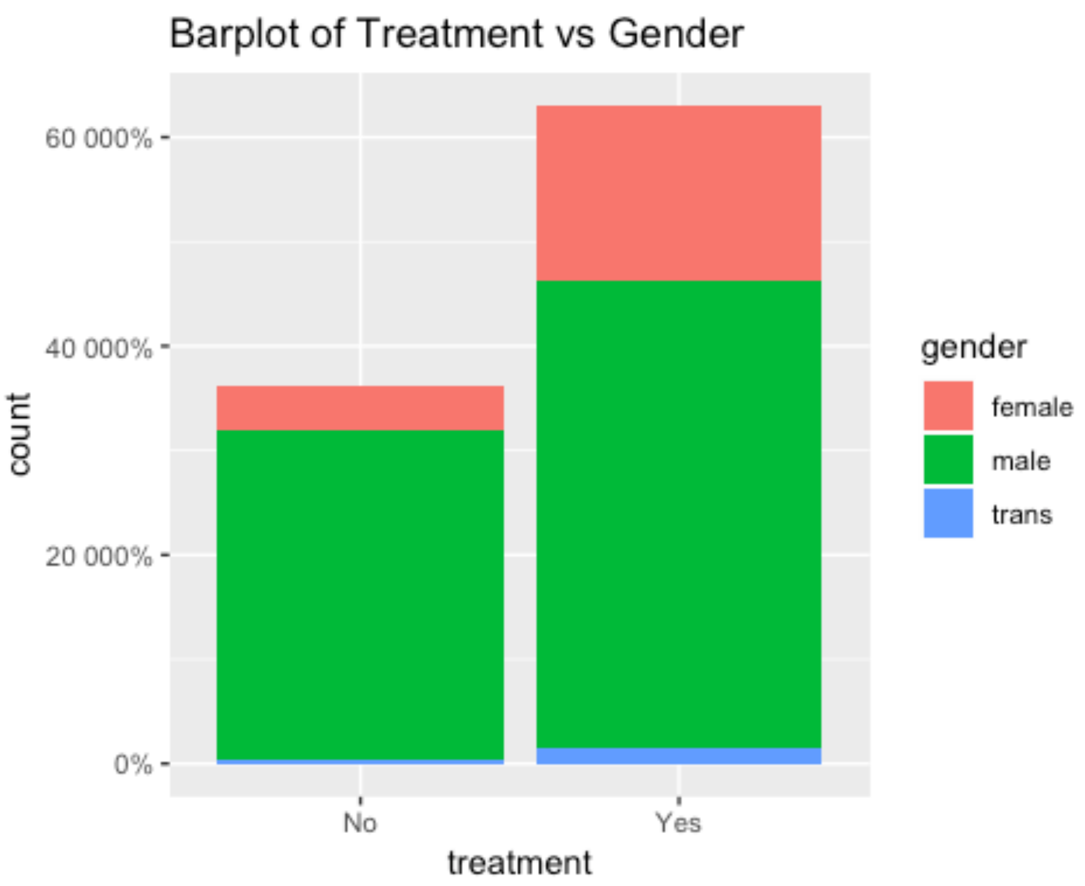
If someone knows that there are care options they are more likely to get treatment. The other factors for care options are split pretty close to 50/50 on both no treatment and yes treatment.

```
ggplot(data = data)+geom_bar(aes(x = treatment,fill= anonymity))+
  scale_y_continuous(labels = scales::percent)+
  ggtitle(label = "Barplot of Treatment vs Anonymity")
```



If they know their anonymity is protected they are more likely to get treatment. But, the same thing can be said if they don't know their anonymity.

```
ggplot(data = data)+geom_bar(aes(x = treatment,fill= gender))+
  scale_y_continuous(labels = scales::percent)+
  ggtitle(label = "Barplot of Treatment vs Gender")
```



A larger percentage females are getting treatment than males. It can also be noted that everyone who identified as trans on the survey are getting mental health treatment.

Logistic/Probit Classification

I'm choosing my models by adding predictors(x variables) and seeing which ones are significant and help the model become accurate. If a variable is not significant (alpha = .05) I will drop the variable

```
glm.fit1=glm(data$treatment~data$family_history,data=data,family=binomial)
summary(glm.fit1)
```

```
##
## Call:
## glm(formula = data$treatment ~ data$family_history, family = binomial,
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8220  -1.1650   0.6495   1.1899   1.1899
##
## Coefficients:
```

```
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -0.02931    0.08560  -0.342    0.732
## data$family_historyYes  1.47830    0.14783  10.000   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1302.8  on 992  degrees of freedom
## Residual deviance: 1191.7  on 991  degrees of freedom
## AIC: 1195.7
##
## Number of Fisher Scoring iterations: 4
```

```
#Confusion matrix

glm1.probs = predict(glm.fit1,data,type = "response")
glm1.pred = rep("No Treatment",993)
glm1.pred[glm1.probs>.5]="Treatment"
table(glm1.pred,data$treatment)
```

```
##
## glm1.pred       No Yes
##   No Treatment 277 269
##   Treatment     85 362
```

```
# error rate
(85 + 269)/993
```

```
## [1] 0.3564955
```

In the first model I only choose family history as my variable. This variable was significant, but the error rate was pretty high, 35.65%. Adding more x variables will make my model more accurate

```
glm.fit2=glm(data$treatment~data$family_history + data$work_interfere,data=da
ta,family=binomial)
summary(glm.fit2)
```

```
##
## Call:
## glm(formula = data$treatment ~ data$family_history + data$work_interfere,
##     family = binomial, data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2063  -0.4882   0.5564   0.7072   2.0911
##
## Coefficients:
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                      -2.0672     0.2064 -10.018  < 2e-16 ***
## data$family_historyYes         1.0836     0.1674   6.472 9.65e-11 ***
## data$work_interfereOften       3.3256     0.3131  10.621  < 2e-16 ***
## data$work_interfereRarely      2.4934     0.2637   9.456  < 2e-16 ***
## data$work_interfereSometimes   2.7711     0.2303  12.033  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1302.80  on 992  degrees of freedom
## Residual deviance:  960.04  on 988  degrees of freedom
## AIC: 970.04
##
## Number of Fisher Scoring iterations: 4
```

```
#confusion matrix
glm2.probs = predict(glm.fit2,data,type = "response")
glm2.pred = rep("No Treatment",993)
glm2.pred[glm2.probs>.5]="Treatment"
table(glm2.pred,data$treatment)
```

```
##
## glm2.pred       No Yes
##   No Treatment 183  30
##   Treatment    179 601
```

```
# error rate
(179 + 30)/993
```

```
## [1] 0.2104733
```

In my second model, I added the variable for work interference. In the model all the variables are significant and my error rate went down, it is now only 21.10%

```
glm.fit3=glm(data$treatment~data$family_history + data$work_interfere + data$
benefits,data=data,family=binomial)
summary(glm.fit3)
```

```
##
## Call:
## glm(formula = data$treatment ~ data$family_history + data$work_interfere +
##     data$benefits, family = binomial, data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4468  -0.6160   0.4871   0.6881   2.2997
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     -2.5705     0.2474 -10.388  < 2e-16 ***
## data$family_historyYes           1.0045     0.1712   5.868 4.42e-09 ***
## data$work_interfereOften       3.4253     0.3225  10.621  < 2e-16 ***
## data$work_interfereRarely      2.4825     0.2705   9.176  < 2e-16 ***
## data$work_interfereSometimes   2.8862     0.2395  12.050  < 2e-16 ***
## data$benefitsNo                0.2125     0.2019   1.053    0.293    
## data$benefitsYes               1.0827     0.2050   5.281 1.28e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 1302.80  on 992  degrees of freedom
## Residual deviance:  926.99  on 986  degrees of freedom
## AIC: 940.99
## 
## Number of Fisher Scoring iterations: 4
```

```
glm3.probs = predict(glm.fit3,data,type = "response")
glm3.pred = rep("No Treatment",993)
glm3.pred[glm3.probs>.5]="Treatment"
table(glm3.pred,data$treatment)
```

```
## 
## glm3.pred       No Yes
##   No Treatment 199  38
##   Treatment    163 593
```

```
# error rate
(163 + 38)/993
```

```
## [1] 0.2024169
```

In the third model, I added the variable for benefits, the factor for benefits yes was signifcant but not the one for benefits no. But it did help the error rate go down to 20.24% so this variable helps to model and will be kept.

```
glm.fit4=glm(data$treatment~data$family_history + data$work_interfere + data$
benefits + data$care_options,data=data,family=binomial)
summary(glm.fit4)
```

```
## 
## Call:
## glm(formula = data$treatment ~ data$family_history + data$work_interfere + 
##     data$benefits + data$care_options, family = binomial, data = data)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -2.5655  -0.5276   0.4430   0.7089   2.3131  
## 
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)    
## (Intercept)                    -2.49635    0.26723  -9.342  < 2e-16 ***
## data$family_historyYes        1.01918    0.17280   5.898 3.68e-09 ***
## data$work_interfereOften      3.39855    0.32448  10.474  < 2e-16 ***
## data$work_interfereRarely     2.41782    0.27201   8.889  < 2e-16 ***
## data$work_interfereSometimes  2.83660    0.24063  11.788  < 2e-16 ***
## data$benefitsNo              -0.08447    0.22383  -0.377  0.70590    
## data$benefitsYes              0.65238    0.23607   2.764  0.00572 ** 
## data$care_optionsNot sure    -0.10740    0.21813  -0.492  0.62246    
## data$care_optionsYes          0.67924    0.21509   3.158  0.00159 ** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 1302.80  on 992  degrees of freedom
## Residual deviance:  914.35  on 984  degrees of freedom
## AIC: 932.35
## 
## Number of Fisher Scoring iterations: 5
```

```
glm4.probs = predict(glm.fit4,data,type = "response")
glm4.pred = rep("No Treatment",993)
glm4.pred[glm4.probs>.5]="Treatment"
table(glm4.pred,data$treatment)
```

```
## 
## glm4.pred       No Yes
##   No Treatment 207  41
##   Treatment    155 590
```

```
# error rate
(155 + 41)/993
```

```
## [1] 0.1973817
```

This model added the variable for care options like the third model only one of the coeffcients(Care_options;yes) was significant. But the error rate went down to 19.74% so this variable will be kept.

```
glm.fit5=glm(data$treatment~data$family_history + data$work_interfere + data$
benefits + data$care_options + data$anonymity,data=data,family=binomial)
summary(glm.fit5)
```

```
## 
## Call:
## glm(formula = data$treatment ~ data$family_history + data$work_interfere + 
##     data$benefits + data$care_options + data$anonymity, family = binomial, 
##     data = data)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -2.4933  -0.5116   0.4180   0.7152   2.3310  
```

```
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    -2.5288     0.2694  -9.386  < 2e-16 ***
## data$family_historyYes          1.0290     0.1736   5.927 3.08e-09 ***
## data$work_interfereOften        3.4128     0.3260  10.469  < 2e-16 ***
## data$work_interfereRarely       2.4413     0.2736   8.922  < 2e-16 ***
## data$work_interfereSometimes    2.8402     0.2414  11.767  < 2e-16 ***
## data$benefitsNo                -0.1076     0.2268  -0.474  0.63522
## data$benefitsYes                0.5612     0.2409   2.330  0.01982 *
## data$care_optionsNot sure      -0.1197     0.2195  -0.545  0.58560
## data$care_optionsYes            0.5884     0.2212   2.659  0.00783 **
## data$anonymityNo               -0.1404     0.3539  -0.397  0.69151
## data$anonymityYes               0.3756     0.2098   1.790  0.07338 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1302.80  on 992  degrees of freedom
## Residual deviance:  910.61  on 982  degrees of freedom
## AIC: 932.61
##
## Number of Fisher Scoring iterations: 5
```

```
glm5.probs = predict(glm.fit5,data,type = "response")
glm5.pred = rep("No Treatment",993)
glm5.pred[glm5.probs>.5]="Treatment"
table(glm5.pred,data$treatment)
```

```
##
## glm5.pred       No Yes
##   No Treatment 203  38
##   Treatment    159 593
```

```
#error rate
(159 + 38)/993
```

```
## [1] 0.1983887
```

This model I added the variable for anonymity. This variable was not significant and the error rate went up slightly to 19.83%. This variable will not be kept.

```
glm.fit6=glm(data$treatment~data$family_history + data$work_interfere + data$
benefits + data$care_options + data$gender ,data=data,family=binomial)
summary(glm.fit6)
```

```
##
## Call:
## glm(formula = data$treatment ~ data$family_history + data$work_interfere +
##     data$benefits + data$care_options + data$gender, family = binomial,
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5232  -0.5105   0.4172   0.7227   2.3611
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   -1.89660    0.33229  -5.708 1.15e-08 ***
## data$family_historyYes         0.96685    0.17395   5.558 2.73e-08 ***
## data$work_interfereOften       3.37877    0.32691  10.335  < 2e-16 ***
## data$work_interfereRarely      2.39376    0.27483   8.710  < 2e-16 ***
## data$work_interfereSometimes  2.80906    0.24258  11.580  < 2e-16 ***
## data$benefitsNo               -0.05139    0.22514  -0.228  0.81946
## data$benefitsYes               0.61570    0.23688   2.599  0.00934 **
## data$care_optionsNot sure     -0.15718    0.21976  -0.715  0.47447
## data$care_optionsYes           0.64592    0.21636   2.985  0.00283 **
## data$gendermale               -0.67006    0.22671  -2.956  0.00312 **
## data$gendertrans              -0.01388    0.69316  -0.020  0.98403
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1302.80  on 992  degrees of freedom
## Residual deviance:  904.42  on 982  degrees of freedom
## AIC: 926.42
##
## Number of Fisher Scoring iterations: 5
```

```r
glm6.probs = predict(glm.fit6,data,type = "response")
glm6.pred = rep("No Treatment",993)
glm6.pred[glm6.probs>.5]="Treatment"
table(glm6.pred,data$treatment)
```

```
##
## glm6.pred       No Yes
##   No Treatment 202  38
##   Treatment    160 593
```

```r
#error rate
(160 + 38) / (993)
```

```
## [1] 0.1993958
```

For the final model, I dropped the anonmity variable and replaced it with gender. The variable for gender(male) was significant but, the error rate went up to 19.94 with the addition of the variable so it will be dropped.

The best model was model 4,this model had the lowest error rate of 19.73% and will be my final model i choose. In model 4 all the variables were signficant. In Model 4, although all the the variables are significant. the ones that affect the model the most are the work_interfer factors because they all have the highest coeffcients. All their coeffcients are greater than 1.Family history is the next signifcant model because it's coeffcient is about 1. The other variables are signifcant but are less important to the model. They help the model become more accurate.

True positive and negative rate of model 4

```
glm.probs=predict(glm.fit4,type="response")
glm.pred=predict(glm.fit4)

glm.pred=rep("No Treatment",993)
glm.pred[glm.probs>.5]="Treatment"

#confusion matrix

table(glm.pred,data$treatment)

##
## glm.pred        No Yes
##   No Treatment 207  41
##   Treatment    155 590
```

True positive rate, the percentage of true Treatment predictions

```
590 / (590 + 155)

## [1] 0.7919463
```

True negative rate, the percentage of true No treatment predictions

```
207 /(207 + 41)

## [1] 0.8346774
```

overall percentage of correct predictions

```
(207 + 590)/993

## [1] 0.8026183
```

This says that my model is okay at predicting whether or not someone is going to recieve mental health treatment. The overall amount of correct predictions is about 80%. this is better than guessing which would have a correct percentage of around 50%.

Probit Regression

```
glm.fits=glm(data$treatment~data$family_history + data$work_interfere + data$
benefits + data$care_options,family = binomial(link = "probit"))
glm.probs=predict(glm.fits,type="response")
glm.pred=rep("No Treatment",993)
glm.pred[glm.probs>.5]="Treatment"
table(glm.pred,data$treatment)

##
## glm.pred        No Yes
##   No Treatment 198  36
##   Treatment    164 595

#true prediton rate
(198 + 595)/993

## [1] 0.7985901
```

Probit has a slightly lower accurate prediction rate than Logistic regression.

Ridge

```
#divide data into train and test
set.seed(101)
n <- nrow(data)
data.index <- sample(1:n , size=round(n*0.7))
train1 <- data[data.index,]
test1 <- data[-data.index,]

x_train = model.matrix(treatment~., train1)[,-1]
x_test = model.matrix(treatment~., test1)[,-1]
y_train = train1$treatment
y_test = test1$treatment
```

Ridge

```
library(glmnet)
library(dplyr)
x = model.matrix(treatment~., data)[,-1] # trim off the first column
                                         # leaving only the predictors
y = data$treatment

grid = 10^seq(10, -2, length = 100)
ridge_mod = glmnet(x, y, alpha = 0, lambda = grid,family = "binomial")

# train your model on the training set, remember that we set alpha to zero fo
r ridge and to one for lasso
ridge_mod = glmnet(x_train, y_train, alpha=0, lambda = grid,family = "binomia
l")
# predict using the trained model and your X_test set
ridge_pred = predict(ridge_mod, s = 4, newx = x_test)
# calculate MSE
ridge_pred <- ifelse(ridge_pred < .5,0,1)
y_test <- ifelse(y_test=="Yes",1,0)
mean((ridge_pred != y_test))

## [1] 0.2852349
```

The initial error rate of the ridge model is 28.52 percent, I will use cross validation to find the flexibly of the model(lambda)

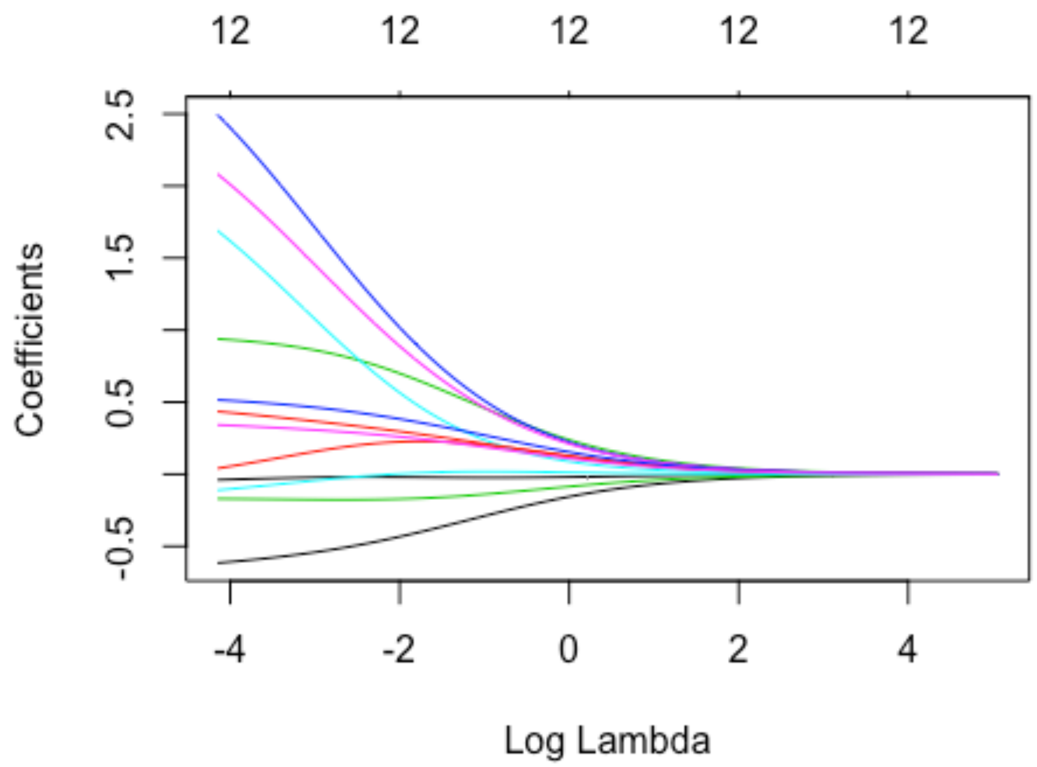
```
#using cross validation to find the lambda witin 1 se of the minimum lambda
set.seed(1)
cv.out = cv.glmnet(x_train, y_train, alpha = 0,family= "binomial") # Fit ridg
e regression model on training data
bestlam = cv.out$lambda.1se  # Select lamda that minimizes training MSE
bestlam

## [1] 0.05696204
```

The value of the best lambda is .0569. This lambda will minimize the Error rate.

```
ridge_pred = predict(ridge_mod, s = bestlam, newx = x_test) # Use best lambda
to predict test data
ridge_pred <- ifelse(ridge_pred <.5,0,1)
mean(ridge_pred != y_test) # Calculate test error rate

## [1] 0.261745
```

The error rate with the new lambda. The error rate went from 28.52 percent down to 26.17 percent.

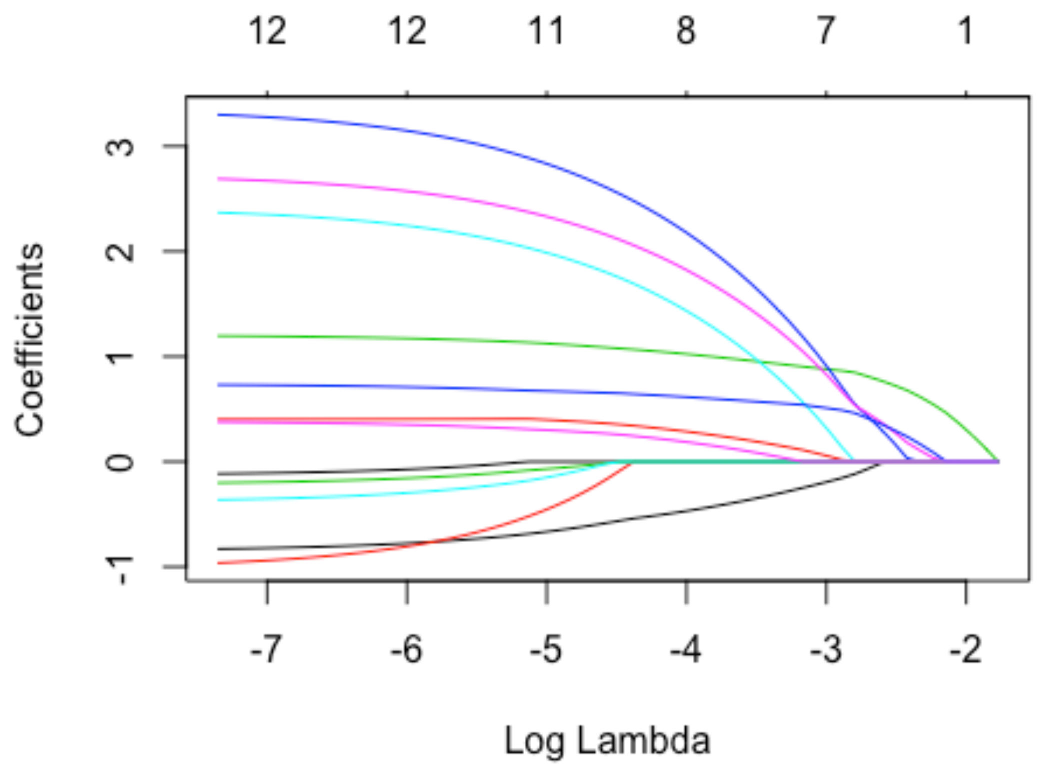
```
out = glmnet(x, y, alpha = 0,family = "binomial") # Fit ridge regression mode
l on the FULL dataset (train and test)
#predict(out, type = "coefficients", s = bestlam)[1:13,] # Display coefficien
ts using lambda chosen by CV

plot(out, xvar = "lambda")
```



The coeffcients get close to 0 as log lambda increases but never quite hit 0. This is what makes ridge different from lasso.

Lasso

```
#Lasso
# We are going to use the same train and test sets that we produced earlier
lasso_mod = glmnet(x_train,
                   y_train,
                   alpha = 1,family = "binomial") # Fit lasso model on traini
ng data
ls(lasso_mod)

##  [1] "a0"          "beta"        "call"        "classnames" "dev.ratio"
##  [6] "df"          "dim"         "jerr"        "lambda"     "nobs"
## [11] "npasses"     "nulldev"     "offset"

plot(lasso_mod,  xvar = "lambda")    # Draw plot of coefficients vs your lamb
da parameter
```



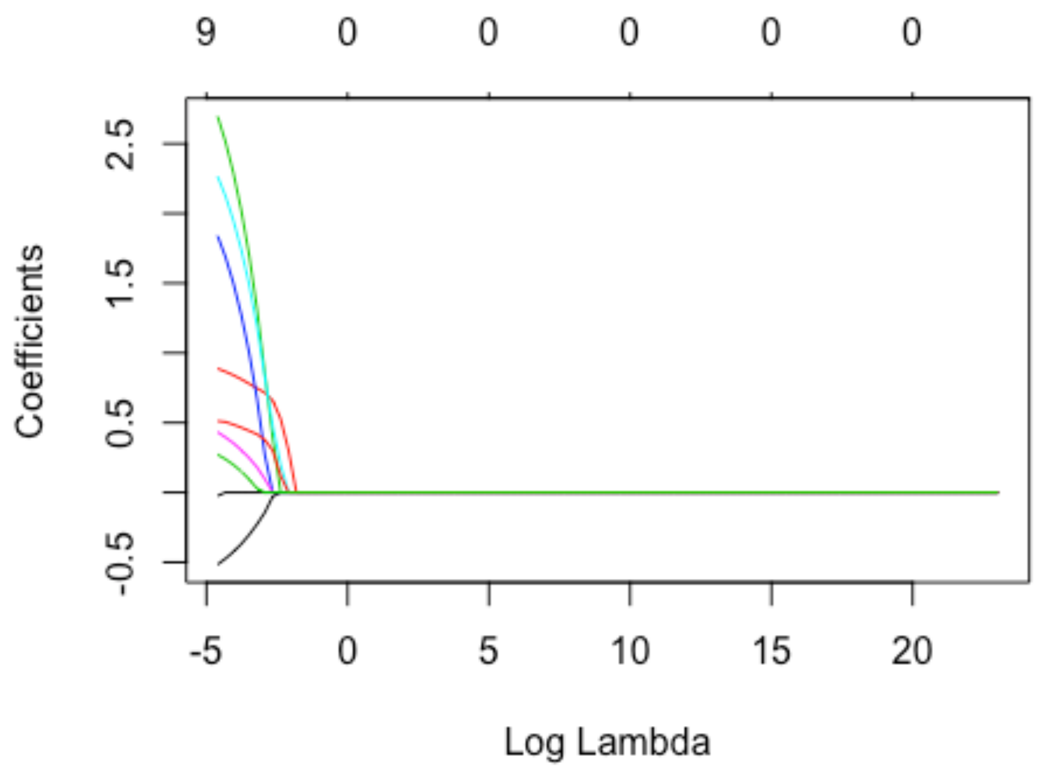
In lasso the coeffcients go to 0 as lambda increases.

```
set.seed(1)
cv.out = cv.glmnet(x_train, y_train, alpha = 1,family= "binomial") # Fit lass
o model on training data



bestlam = cv.out$lambda.1se # Select lamda that minimizes training MSE
bestlam

## [1] 0.02409063

lasso_pred = predict(lasso_mod, s = bestlam, newx = x_test) # Use best lambda
to predict test data
lasso_pred <- ifelse(lasso_pred < 0.5,0,1)
mean(lasso_pred != y_test) # Calculate test error rate

## [1] 0.2583893
```

In lasso the best lambda is 0.024. This lambda minimizes the test error to 25.83 percent.

```
out = glmnet(x, y, alpha = 1, lambda = grid,family = "binomial") # Fit lasso
model on full dataset
lasso_coef = predict(out, type = "coefficients", s = bestlam)[1:13,] # Displa
```

```
y coefficients using lambda chosen by CV
#lasso_coef

plot(out, xvar = "lambda")
```

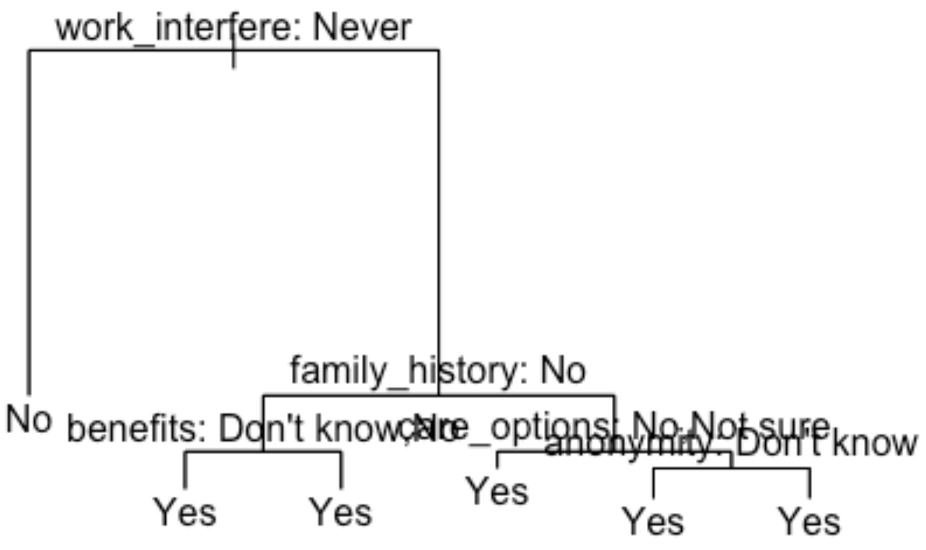


The coeffcients in Lasso all go to 0 as log of lambda increases. In ridge they never reach 0 but in lasso because alpha is 1 they reach 0.

Decision Tree

```
library(tree)
library(ISLR)
library(dplyr)
library(ggplot2)

tree_health = tree(treatment ~ .  -treatment, train1)
summary(tree_health)

##
## Classification tree:
## tree(formula = treatment ~ . - treatment, data = train1)
## Variables actually used in tree construction:
## [1] "work_interfere" "family_history" "benefits"       "care_options"
## [5] "anonymity"
## Number of terminal nodes:  6
## Residual mean deviance:  0.9071 = 625 / 689
## Misclassification error rate: 0.2086 = 145 / 695
```

```
plot(tree_health)
text(tree_health, pretty = 0)
```



This tree is the initial tree that was made. It is not pruned yet which is why many of the terminal nodes end with (Yes and Yes). The starting and most important variable in this tree is The factor for work interfence:Never. So if someone feels like their mental health never interfers with their work they will not get treatment. Every other option says they will get treatment.

```
tree_pred = predict(tree_health, test1, type = "class")
table(tree_pred, test1$treatment)
```

```
## 
## tree_pred  No Yes
##       No   55   8
##       Yes  56 179
```

```
#Accuracy rate
(55+179)/(298)
```

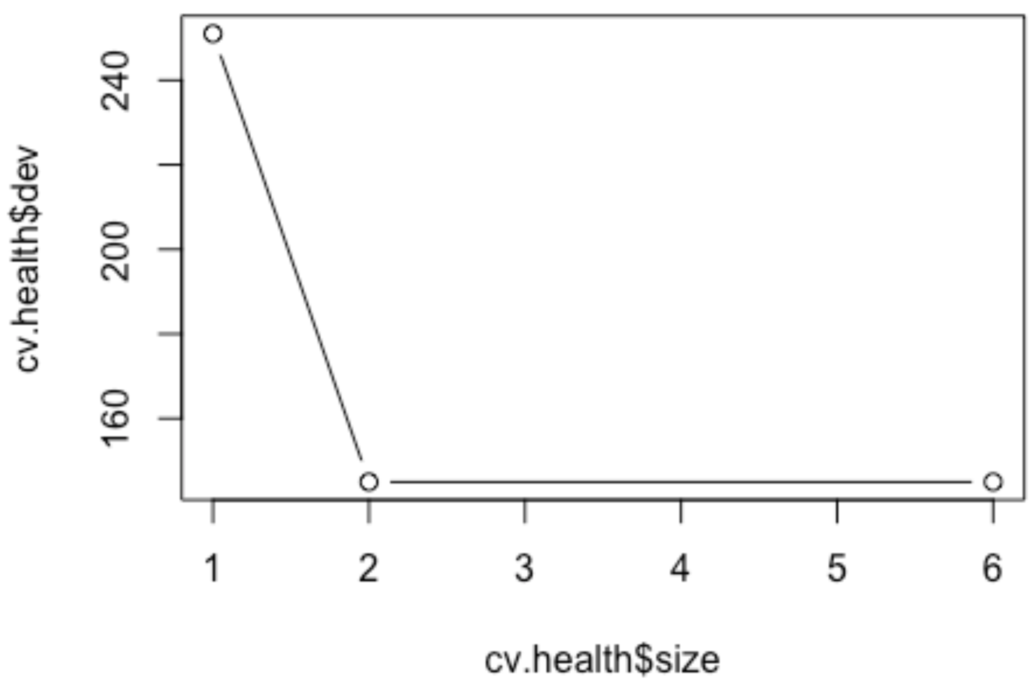
```
## [1] 0.7852349
```

The accuracy rate of this model is 78.52 percent

```
#error rate
1-((55+179)/(298))

## [1] 0.2147651
```

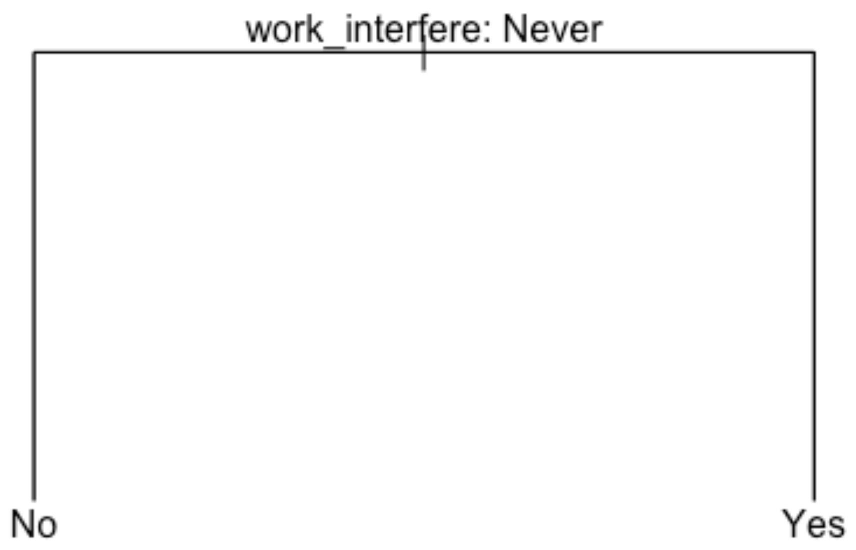
Error rate is 21.47%

```
#pruning
set.seed(3)
cv.health = cv.tree(tree_health, FUN = prune.misclass)
plot(cv.health$size, cv.health$dev, type = "b")
```



According to the prune and its graph in order to minimize the errors the amount of nodes should be 2.

```
prune_health = prune.misclass(tree_health, best = 2)
plot(prune_health)
text(prune_health, pretty = 0)
```



In the pruned tree the only variable that is relevant is work interference:Never. For this tree if someone feels like their mental health never interfers with their work they will not get treatment. If they don't agree with that statement then they will get treatment

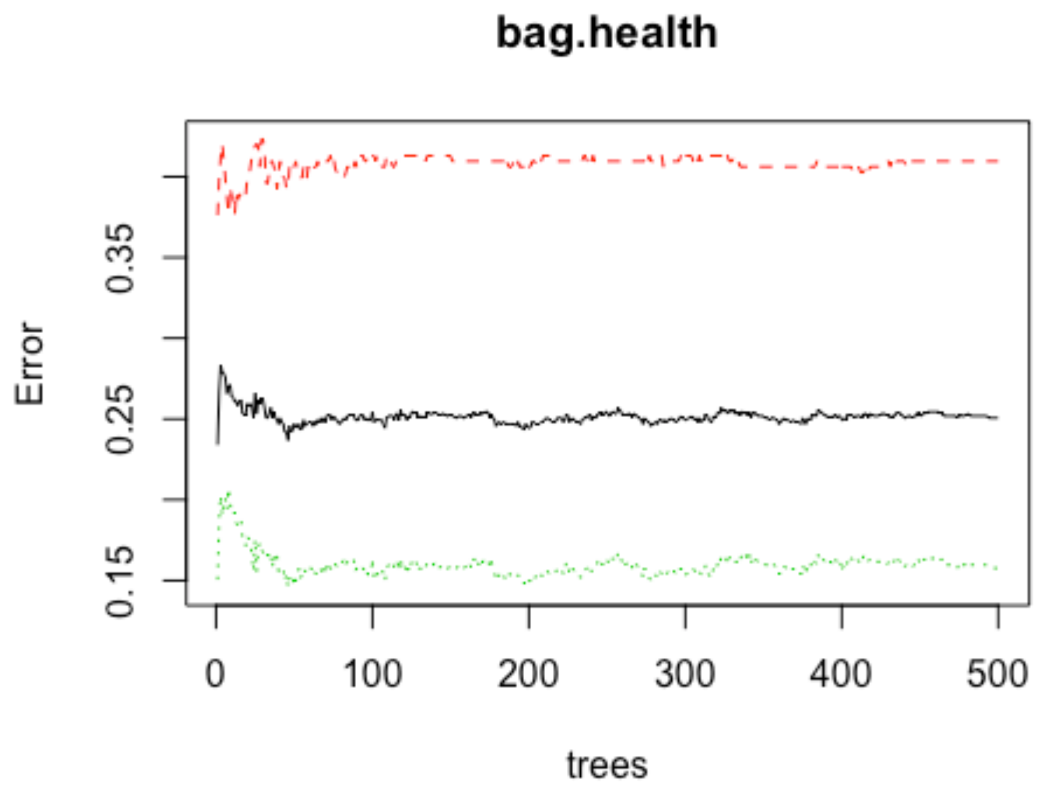
```
tree_pred = predict(prune_health, test1, type = "class")
table(tree_pred, test1$treatment)

##
## tree_pred  No Yes
##       No   55   8
##       Yes  56 179

#error rate
1-((55+179)/298)

## [1] 0.2147651
```

The pruned tree has the same error rate as the orginal tree but this tree gets rid of the parts of the tree that were not relevant. (The ones that nodes ended with Yes as both options)

Oddly enough this is more accuarte than the Lasso and Ridge models. It might be because this model uses a variable that is known to be very signifcant in predicting treatment. The Lasso and Ridge models use all variables,even the ones that aren't signifcant.

Bagging

```
# dividing data into train and test
set.seed(101)
n <- nrow(data)
data.index <- sample(1:n , size=round(n*0.8))
train1 <- data[data.index,]
test1 <- data[-data.index,]

#2a-b
set.seed(1)
bag.health = randomForest(treatment ~., data=train1,
                          mtry=ncol(train1)-1,
                          importance=TRUE)
bag.health

##
## Call:
##  randomForest(formula = treatment ~ ., data = train1, mtry = ncol(train1) 
-      1, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 25.06%
## Confusion matrix:
##      No Yes class.error
## No  173 120   0.4095563
## Yes  79 422   0.1576846

plot(bag.health)
```



From the plot we can see that the error rate does go down as the amount of trees increase. But that effect is much larger when the amount of trees are less than 100. After 100 the error rate stays about the same.

From the error rate table we see that the out of bag error rate is 25.06 percent. This means that about 25 percent of the predictions are not accurate. This is better than guessing but its no where near perfect. From the matrix you can see that it more accurate predicting if people recieve mental health treatment “Yes”. This has a error rate of 15.7 percent. But, for no it has a error rate of 40.96 percent.

```
# 2-d
importance_matrix <- importance(bag.health)
importance_matrix

##                        No       Yes MeanDecreaseAccuracy MeanDecreaseGini
## gender          10.436843  3.959418             9.974456         16.41370
## family_history  25.197348 20.357658            31.523716         18.55392
## work_interfere  68.208170 72.834427            88.044532        119.49455
## benefits         2.750632  8.050502             8.858850         23.56150
## care_options     2.531192 20.344554            20.461167         24.65594
## anonymity       -1.425376 -2.379567            -2.899247         17.40843
```

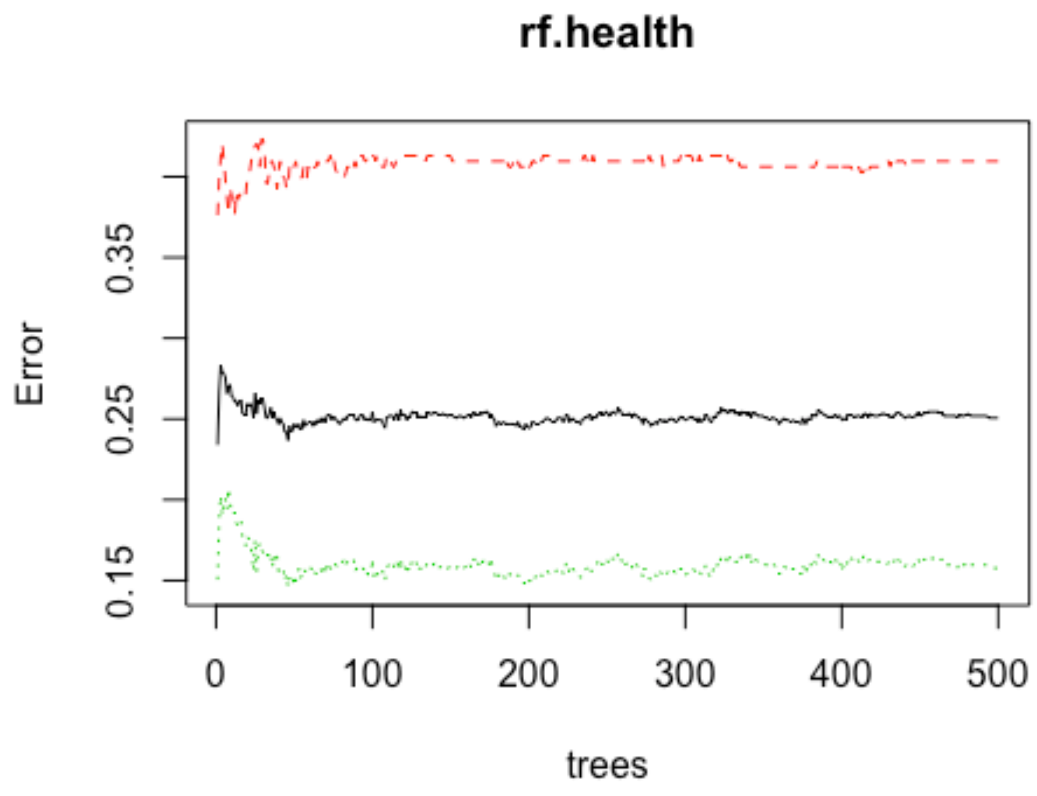
This importance matrix shows that the work_interference variable is the most influential on as it has the highest meanDecreaseAccuracy and the highest meandecrease.

```
#Bagging error rate
train1$treatment_dummy<-2
train1$treatment_dummy[train1$treatment=="No"]<-1
test1$treatment_dummy<-2
test1$treatment_dummy[test1$treatment=="No"]<-1
yhat.bag = predict(bag.health,
                           newdata = test1,
                           type="response")
yhat.bag <- as.numeric(yhat.bag)
yhat.bag <- round(yhat.bag)
(mean(( test1$treatment_dummy != yhat.bag)))

## [1] 0.2110553
```

The error rate for bagging is 21.10 percent.

Random Forest

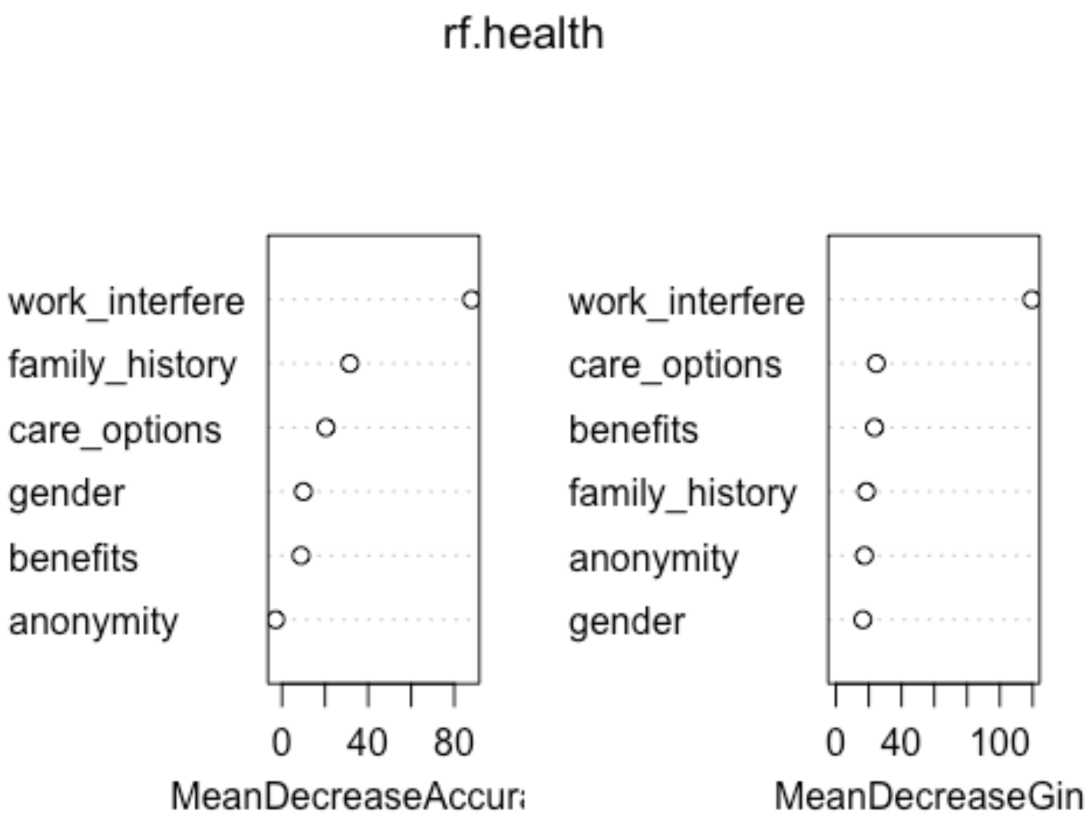
```
#3a/c
set.seed(1)
rf.health = randomForest(treatment~.,
                         data = train1,
                         mtry = 6,
                         importance = TRUE,
                         do.trace = 100)

## ntree      OOB      1      2
##   100:  25.44% 41.30% 16.17%
##   200:  24.43% 40.61% 14.97%
##   300:  24.94% 41.30% 15.37%
##   400:  24.94% 40.61% 15.77%
##   500:  25.06% 40.96% 15.77%

plot(rf.health)
```



Like in the bagging classification, the error rate in random forest stays the same once the amount of trees increase above 100. At 0 trees the error rate is very high but then it lowly levels out to the same numbers.

The error rate table shows that the OOB error stays around 25 percent as the amount of trees increase. In the plot it is the black line. The error rate for 1 which represents how well the model predicts if someone did not recieve mental health treatment is about 40 percent. This is the red line on the plot and the error rate for 2 which represents if someone did recieve mental health treatment is about 15 percent.

```
varImpPlot(rf.health)
```

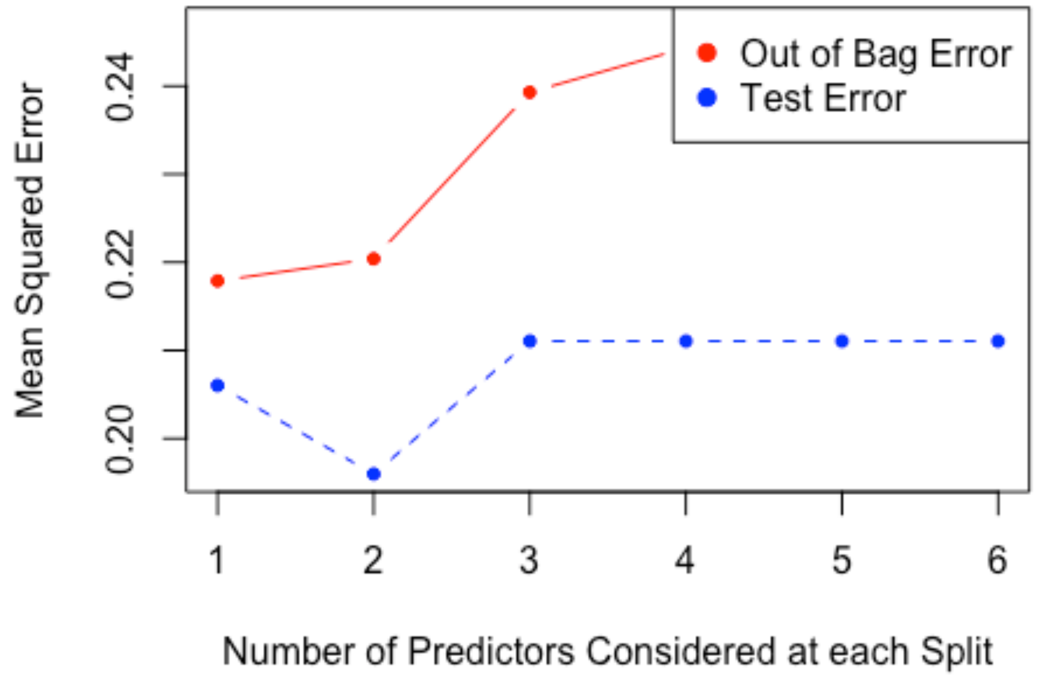


The importance matrix shows that work_interference is the most important variable and anonymity is the least influential variable.

```
rf.oob.err<-double(6)
rf.test.err<-double(6)
for(mtry in 1:6)
{
  rf=randomForest(treatment ~ . , data = train1, mtry=mtry, ntree=400)
  rf.oob.err[mtry] = rf$err.rate[400,1] #Error of all Trees fitted on trainin
g

  pred<-predict(rf,test1) #Predictions on Test Set for each Tree
  rf.test.err[mtry]= mean( (test1$treatment!=pred)) # "Test" Mean Squared Err
or
}
round(rf.test.err,2)

## [1] 0.21 0.20 0.21 0.21 0.21 0.21

round(rf.oob.err,2)

## [1] 0.22 0.22 0.24 0.24 0.25 0.25
```

The oob.err rate for mtry 2 is .20.Also the test error for mtry 2 is .20. This is the best model, because this is where both the errors are minimized.

```
#3f
matplot(1:mtry , cbind(rf.oob.err,rf.test.err), pch=20 , col=c("red","blue"),
type="b",ylab="Mean Squared Error",xlab="Number of Predictors Considered at e
ach Split")
legend("topright",legend=c("Out of Bag Error","Test Error"),pch=19, col=c("re
d","blue"))
```

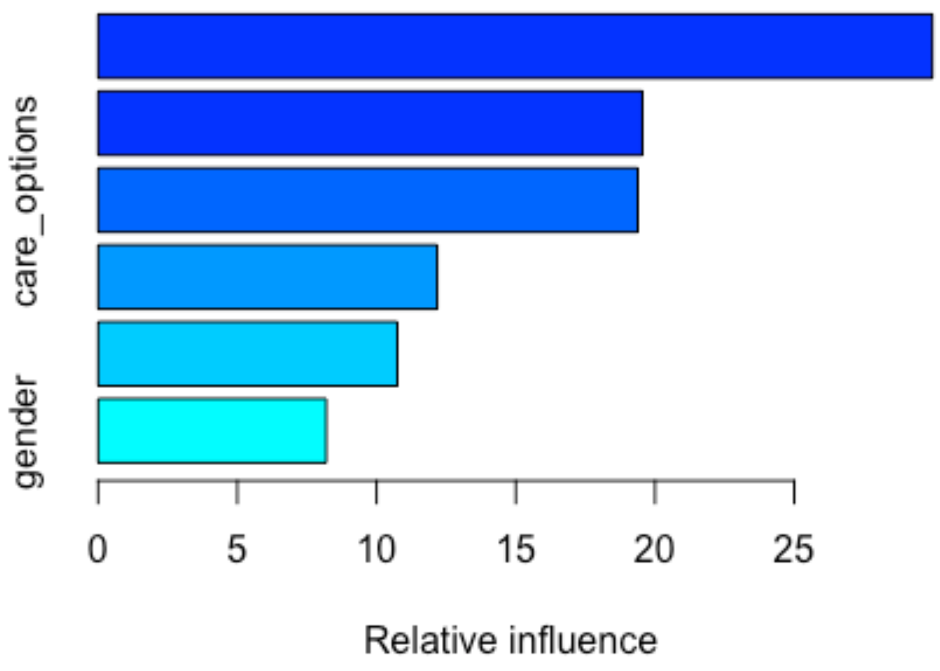


From the plot it can be detertermined that mtry 2 is the one that should be picked because it has a lowest oob error rate and test error. Also it is shown that both the test error and the oob error follow similar patterns.

Boosting

```
## question 4
#4a/c

library(gbm)

## Loaded gbm 2.1.5
```

```
set.seed(1)
train1$treatment_dummy<-1
train1$treatment_dummy[train1$treatment=="No"]<-0
test1$treatment_dummy<-1
test1$treatment_dummy[test1$treatment=="No"]<-0


boost.health = gbm(treatment_dummy~.-treatment,
                   data = train1,
                   distribution = "bernoulli",
                   n.trees = 5000,
                   interaction.depth = 4)
summary(boost.health)
```



```
##                            var   rel.inf
## work_interfere work_interfere 29.958741
## benefits             benefits 19.545260
## care_options     care_options 19.381030
## family_history family_history 12.176962
## anonymity           anonymity 10.744970
## gender                 gender  8.193036
```

From the Importance matrix, like in the other importance matrices is is shown that work_interference is the most important variable. The least important variable is gender in this model

```
# 4b
yhat.boost = predict(boost.health,
                            newdata = test1[,-7],
                            n.trees = 5000,type="response")
yhat.boost<-round(yhat.boost)

(mean(( test1$treatment_dummy != yhat.boost)))

## [1] 0.2261307
```

The error rate is about 22.61 percent for boosting.

XGBoost

```
# question 5 setting up train and test
Y_train_dummy <- 0
library(xgboost)

##
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice

## The following object is masked from 'package:plotly':
##
##     slice

Y_train <- as.matrix(train1[,"treatment"])
X_train <- as.matrix(train1[!names(train1) %in% c("treatment")])
X_train<-X_train[,-7]
X_train<-data.frame(X_train)
X_train<-data.matrix(X_train)

Y_train_dummy[Y_train=="Yes"]<-1
Y_train_dummy[Y_train=="No"]<-0

dtrain <- xgb.DMatrix(data = X_train, label = Y_train_dummy)
X_test <- as.matrix(test1[!names(train1) %in% c("treatment")])
X_test<-X_test[,-7]

# 5a
set.seed(1)
health.xgb = xgboost(data=dtrain,
                     max_depth=5,
                     eta = 0.1,
                     nrounds=100,
                     lambda=0,
                     print_every_n = 10,
                     objective="binary:logistic")
```

```
## [1]  train-error:0.202771
## [11] train-error:0.195214
## [21] train-error:0.195214
## [31] train-error:0.190176
## [41] train-error:0.186398
## [51] train-error:0.185139
## [61] train-error:0.180101
## [71] train-error:0.176322
## [81] train-error:0.177582
## [91] train-error:0.175063
## [100]    train-error:0.176322
```

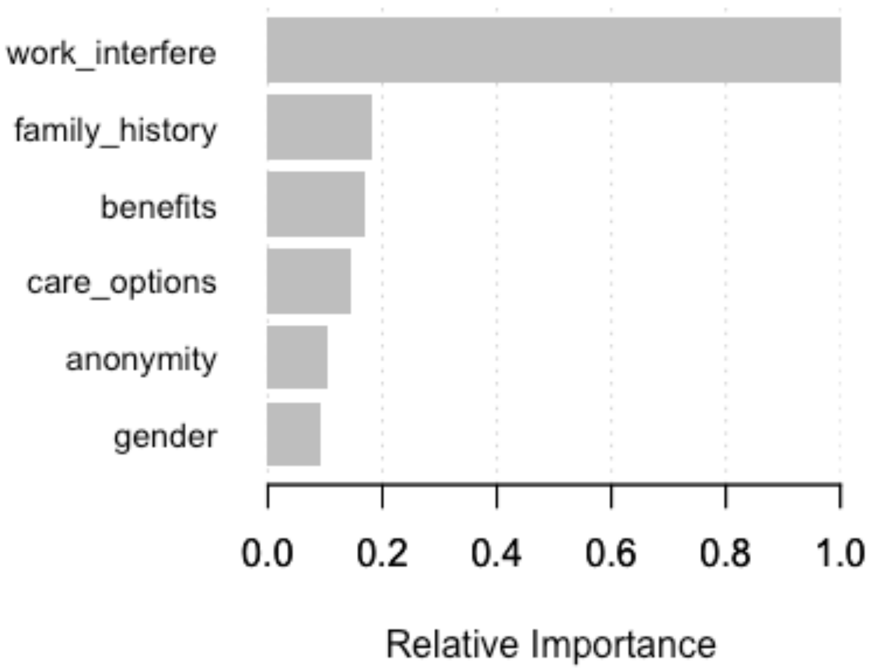
The error table of the train error shows that as the amount of rounds increase the error decreases. It starts out at 20.23 percent and then goes down to 17.63 percent. The more rounds the “smarter” the model gets.

```
#5b
X_test<-data.frame(X_test)
X_test<-data.matrix(X_test)
yhat.xgb <- predict(health.xgb,X_test,type="response")
yhat.xgb<-round(yhat.xgb)
(mean(yhat.xgb != test1$treatment_dummy))
```

```
## [1] 0.2160804
```

The error rate is 21.61 percent in XGBoost

```
# 5c

importance <- xgb.importance(colnames(X_train),model=health.xgb)
xgb.plot.importance(importance, rel_to_first=TRUE, xlab="Relative Importance"
)
```

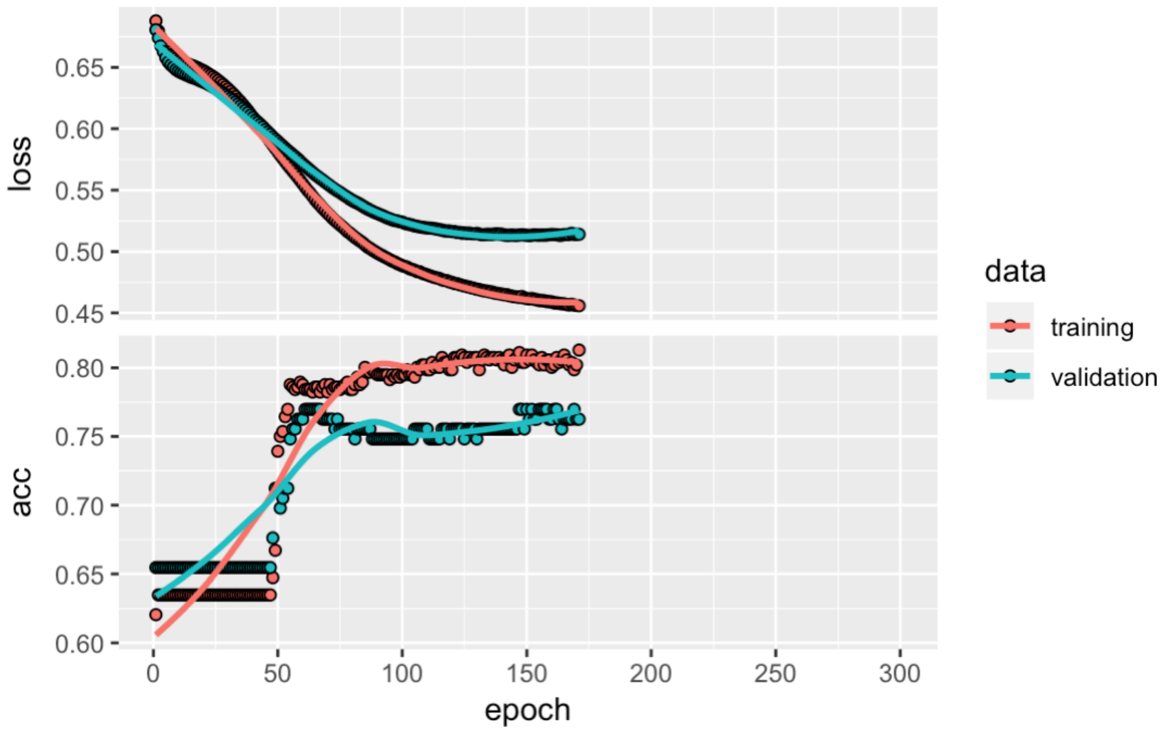


As all the other importance matrices before have stated, work_interfere is the more important variable. But in this model the least important variable is gender.

Neural Net

```r
#NN
library(keras)
library(ISLR)
library(dplyr)
#split data into train and test into categorical variables
train1$treatment_dummy2 <- ifelse(train1$treatment == "Yes",1,0)
test1$treatment_dummy2 <- ifelse(test1$treatment == "Yes",1,0)
train1$gender <- as.numeric(train1$gender)
train1$family_history <- as.numeric(train1$family_history)
train1$work_interfere <- as.numeric(train1$work_interfere)
train1$benefits <- as.numeric(train1$benefits)
train1$care_options <- as.numeric(train1$care_options)
train1$anonymity <- as.numeric(train1$anonymity)

test1$gender <- as.numeric(test1$gender)
test1$family_history <- as.numeric(test1$family_history)
test1$work_interfere <- as.numeric(test1$work_interfere)
test1$benefits <- as.numeric(test1$benefits)
```

```r
test1$care_options <- as.numeric(test1$care_options)
test1$anonymity <- as.numeric(test1$anonymity)

train_labels <- to_categorical(train1[,"treatment_dummy2"],2)
train_data <- as.matrix(train1[!names(train1) %in% c("treatment","treatment_d
ummy2")])
test_data <- as.matrix(test1[!names(train1) %in% c("treatment","treatment_dum
my2")])
test_labels <- to_categorical(test1[,"treatment_dummy2"])

model <- keras_model_sequential() %>%
  layer_dense(units = 64, activation = "softmax",
              input_shape = dim(train_data)[2]) %>%
  layer_dense(units = 64, activation = "softmax") %>%
  layer_dense(units = 2, activation= "softmax")

model %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_rmsprop(),
  metrics = c('accuracy')
)

early_stop <- callback_early_stopping(monitor = "val_loss", patience = 20)

epochs=300
history_class <- model %>% fit(
  train_data,
  train_labels,
  epochs = epochs,
  validation_split = 0.2,
  callbacks = list(early_stop)
)

plot(history_class, rep="best")
```



```
test_predictions <- model %>% predict(test_data)

test_class <- model %>% predict_classes(test_data)

table(test_labels[,2], test_class)
```

```
    test_class
       0   1
  0   61  50
  1   20 167
```

```
#misclassification rate
(20+50)/(61+50+20+167)

## [1] 0.2348993
```

The misclassification rate/ error rate in Neural Network is 23.94 Percent.

Comparing all models / Conclusion

The error rate for logistic regression was 19.74 percent.

The error rate for Lasso was 25.83 percent

The error rate for Ridge was 26.17 percent

The error rate for Decision tree was 21.47 percent

The error rate for bagging was 21.10 percent.

The error rate for random forest was 20 percent.

The error rate for boosting was 22.61 percent.

The error rate for xg boost was 21.61 percent.

The error rate for Neural Networks was 23.48 percent.

The overall best error rate was my logistic regression model. I believe this was the case because for this model I went through and got rid of variables that were not significant. The machine learning models all had some x variables that weren't significant and could of made the error rate go up slightly. The shrinkage methods (Lasso and Ridge) performed the worst. I think the method of minimizing the coeffcients to 0,or close to 0 did not help with my data because my data contained mostly factor variables. My decsion tree was accurate but when it was pruned it only contained two nodes. This tree also showed that work interference was the most influential of the x variables. The Neural network performed ok, this was the most complicated method, so it was harder to minimize the errors. The best error rate from methods that used machine learning was random forest because I used mtry to find the best one to use. But the other 3 error rates are around the same. In fact the bagging and XG boost ones are exactly the same and boosting is only 2 percent higher. Overall these models are not perfect, but are much better than guessing. The most influential variable in all the models was work_interference. This makes sense because the work interference variable was the variable that represented if people thought their mental health directly affected their work. The least influential variable was either gender or the ablity to remain anon if you wanted to reach out for help. This was a bit shocking because I assumed that men would be less likely to get help and that people would value staying anon. Overall,most of the models had an error rate in the early 20's.The models that performed the best were random forest and the logistic regression. I think the best conclusion I can get from running all of these models, is that the variable work_interfere is the most influential in order to see if someone will get mental health treatment. I think this is important for work places to know that at the end of the day,no matter how many resources a work place has the most important factor in determining if someone will get mental health help is if they feel like their mental health is directly affecting their ablilty to work. The least influential variable was gender. The fact that gender was least influential was suprising to me because I thought that men would be less likly to seek help.