Final - Predicting Opioid Abuse

Deanna Schneider

July 9, 2018

```
pacman::p_load(caret,ggplot2,ggpubr,RColorBrewer,grid,gridExtra,tree,randomForest,nnet,parallel,doParallel,e1071,pR
OC,C50,mlbench,plyr,NeuralNetTools,ggsci,scales,kableExtra,lemon,pander,glmnet,missForest,plyr,DescTools,ggcorrplot
,psych,rcompanion,randomForestExplainer)
```

Predicting Opioid Abuse from Perception of Risk

Opioids are fast becoming an epidemic in Wisconsin. University of Wisconsin Extension has launched an opioid crisis response team, which is attempting to understand the nature of opioid misuse and abuse and is in the process of developing educational content to address opioid abuse awareness and prevention. UW-Extension teaches adults on many topics - from parenting to healthy food choices to family financial management. In many of our existing programs, it would be possible to include additional content aimed at opioid abuse prevention. However, doing so will often mean that another topic may need to be removed. If we could identify whether a particular audience of learners is more at risk for opioid abuse using simple, non-intrusive questions, we could use that information in curriculum planning.

The data for this analysis was taking from the National Survey on Drug Use and Health 2016, available on kaggle.com (https://www.kaggle.com/cheul0/nsduh-2016) and at the Substance Abuse and Mental Health Services Administration site (http://datafiles.samhsa.gov/study-dataset/national-survey-drug-use-and-health-2016-nsduh-2016-ds0001-nid17185).

```
#read in the full dataset
nsduh <- read.csv('NSDUH_2016_Tab.csv')

#only look at adults
nsduh <- nsduh[which(nsduh$CATAG6 != 1), ]
dim(nsduh)

## [1] 42625  2664

table(nsduh$OPINMYR)

## 
##     0     1 
## 40244  2381
```

This dataset contains 2664 variables. Of those, many of them are recoded or imputed versions of the base questions used in the survey. Additionally, there are multiple sections to the survey, many of which are quite intrusive, asking for detailed histories about substance use and abuse. One section of the survey, however, deals with opinions about the riskiness of both legal and illegal substances. This section could easily be used in a pretest format with adult learners. For the purposes of this analysis, we will use the risk section and a few selected demographics questions, most of which are already collected about our clientele to meet federal reporting requirements.

```
cols <- c('OPINMYR', #response variable/binary misused opioids in last year
          #beliefs about the riskiness of trying/using drugs and alcohol
          'RSKCIGPKD', 'RSKMRJMON', 'RSKMRJWK', 'RSKLSDTRY', 'RSKLSDWK', 'RSKHERTRY',
          'RSKHERWK', 'RSKCOCMON',
            'RSKCOCWK', 'RSKBNGDLY', 'RSKBNGWK'

            #selected demographics
            ,'CATAG6', 'IRSEX', 'EDUHIGHCAT', 'HEALTH2', 'IRMARIT', 'IRWRKSTAT', 'NEWRACE2'
)

#get just this set of columns
nsduh.subset <- nsduh[, cols]



#review
summary(nsduh.subset)

#get complete cases (this is actually everything - except 3 rows where health had missing data. 3 rows out of this dataset
should not affect overall performance.)
cc <- nsduh.subset [complete.cases(nsduh.subset ), ]

#convert response to factor
cc$OPINMYR <- as.factor(cc$OPINMYR)

#response
levels(cc$OPINMYR) <- c('No', 'Yes')
table(cc$OPINMYR)

cc$isUser <- cc$OPINMYR
cc$OPINMYR <- NULL

#move our response variable to the front
cc <- cc[, c(19, 1:18)]

#review
summary(cc)


#clear up our environment
rm(list=setdiff(ls(), "cc"))
```

Demographic variables

The demographic variables contain a mix of ordered and non-ordered categorical variables. The non-ordered variables include marital status, employment status, sex and race. The ordered variables include age, health status and education. For education, the survey asked for the highest level of schooling. For most individuals, achieving a higher level of schooling would imply that the lower level of schooling was also achieved. The health status variable is a subjective scale ranging from Fair/Poor to Excellent.

```
#ordered factors
cc$CATAG6 <- ordered(cc$CATAG6)
cc$EDUHIGHCAT <- ordered(cc$EDUHIGHCAT)
cc$HEALTH2 <- ordered(cc$HEALTH2)

#nominal factors
cc$IRMARIT <- factor(cc$IRMARIT)
cc$IRSEX <- factor(cc$IRSEX)
cc$IRWRKSTAT <- factor(cc$IRWRKSTAT)
cc$NEWRACE2 <- factor(cc$NEWRACE2)

#check results
str(cc)
```

“Other” types of responses

All the variables for risk contain the options of “Don’t know,” “Refused,” and “Blank.” These make up a small portion of the data. Various approaches were attempted to deal with this data. Ultimately, the approach taken was to convert the “Don’t Know” to a “neutral” number at the middle of the scale, and to convert the “Refused” and “Blank” to NA and impute appropriate replacements.

```
rskCols <- c('RSKCIGPKD', 'RSKMRJMON', 'RSKMRJWK', 'RSKLSDTRY', 'RSKLSDWK', 'RSKHERTRY',
             'RSKHERWK', 'RSKCOCMON', 'RSKCOCWK', 'RSKBNGDLY', 'RSKBNGWK')


for (c in 1:length(rskCols)){
  #get the column number
  thisCol <- rskCols[c]
  cc[[thisCol]][which(cc[[thisCol]] == 94)] <- 2.5
  cc[[thisCol]][which(cc[[thisCol]] > 94)] <- NA

  rLev = c(1,2,2.5,3,4)
  rLab = c(-1,-2,0,1,2)
cc[[thisCol]] <- ordered(cc[[thisCol]], levels=rLev, labels=rLab)
}
summary(cc)

##  isUser      RSKCIGPKD    RSKMRJMON     RSKMRJWK      RSKLSDTRY
##  No :40241   -1  : 1358   -1  :12326   -1  :10211   -1  : 1504
##  Yes: 2381   -2  : 2252   -2  :12478   -2  :11282   -2  : 4919
##              0   :  232   0   :  459   0   :  486   0   :  823
##              1   : 8965   1   : 7999   1   : 9187   1   : 7724
##              2   :29766   2   : 9300   2   :11402   2   :27588
##              NA's:   49   NA's:   60   NA's:   54   NA's:   64
##
##  RSKLSDWK     RSKHERTRY    RSKHERWK     RSKCOCMON    RSKCOCWK
##  -1  :  635   -1  :  482   -1  :  372   -1  :  851   -1  :  417
##  -2  : 1531   -2  : 1359   -2  :  367   -2  : 3747   -2  : 1105
##  0   :  838   0   :  522   0   :  508   0   :  519   0   :  519
##  1   : 5383   1   : 3927   1   : 1499   1   : 7730   1   : 4168
##  2   :34167   2   :36282   2   :39827   2   :29720   2   :36356
##  NA's:   68   NA's:   50   NA's:   49   NA's:   55   NA's:   57
##
##  RSKBNGDLY    RSKBNGWK     CATAG6    IRSEX     EDUHIGHCAT HEALTH2
##  -1  :  570   -1  : 1476   2:13660   1:19852   1: 5488    1: 9920
##  -2  : 3195   -2  : 8048   3: 8751   2:22770   2:11306    2:16141
##  0   :  272   0   :  284   4:11360             3:14435    3:11769
##  1   :10772   1   :15185   5: 5241             4:11393    4: 4792
##  2   :27762   2   :17581   6: 3610
##  NA's:   51   NA's:   48
##
##  IRMARIT   IRWRKSTAT NEWRACE2
##  1:17470   1:22056   1:25967
##  2: 1289   2: 6732   2: 5473
##  3: 4752   3: 2618   3:  643
##  4:19111   4:11216   4:  224
##                      5: 1888
##                      6: 1355
##                      7: 7072

str(cc)

## 'data.frame':    42622 obs. of  19 variables:
##  $ isUser    : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 1 1 1 1 1 ...
##  $ RSKCIGPKD : Ord.factor w/ 5 levels "-1"<"-2"<"0"<..: 5 5 4 5 4 5 4 5 2 4 ...
##  $ RSKMRJMON : Ord.factor w/ 5 levels "-1"<"-2"<"0"<..: 5 2 4 2 4 5 1 1 2 1 ...
##  $ RSKMRJWK  : Ord.factor w/ 5 levels "-1"<"-2"<"0"<..: 4 2 4 2 2 5 2 1 2 1 ...
##  $ RSKLSDTRY : Ord.factor w/ 5 levels "-1"<"-2"<"0"<..: 5 1 2 5 4 5 5 5 4 4 ...
##  $ RSKLSDWK  : Ord.factor w/ 5 levels "-1"<"-2"<"0"<..: 5 4 4 5 5 5 5 5 4 4 ...
##  $ RSKHERTRY : Ord.factor w/ 5 levels "-1"<"-2"<"0"<..: 5 2 5 5 5 5 5 5 5 5 ...
##  $ RSKHERWK  : Ord.factor w/ 5 levels "-1"<"-2"<"0"<..: 5 5 5 5 5 5 5 5 4 5 ...
##  $ RSKCOCMON : Ord.factor w/ 5 levels "-1"<"-2"<"0"<..: 5 2 4 5 5 5 5 5 4 5 ...
##  $ RSKCOCWK  : Ord.factor w/ 5 levels "-1"<"-2"<"0"<..: 5 4 4 5 5 5 5 5 5 5 ...
##  $ RSKBNGDLY : Ord.factor w/ 5 levels "-1"<"-2"<"0"<..: 5 5 5 5 4 5 5 5 2 5 ...
##  $ RSKBNGWK  : Ord.factor w/ 5 levels "-1"<"-2"<"0"<..: 4 4 5 4 4 5 4 5 4 5 ...
##  $ CATAG6    : Ord.factor w/ 5 levels "2"<"3"<"4"<"5"<..: 2 1 3 5 2 4 3 2 1 2 ...
##  $ IRSEX     : Factor w/ 2 levels "1","2": 2 1 1 2 1 1 1 1 1 2 ...
##  $ EDUHIGHCAT: Ord.factor w/ 4 levels "1"<"2"<"3"<"4": 4 3 1 3 4 2 4 2 2 3 ...
##  $ HEALTH2   : Ord.factor w/ 4 levels "1"<"2"<"3"<"4": 1 3 1 2 2 4 2 4 1 2 ...
##  $ IRMARIT   : Factor w/ 4 levels "1","2","3","4": 1 4 1 1 1 2 1 4 4 4 ...
##  $ IRWRKSTAT : Factor w/ 4 levels "1","2","3","4": 4 4 3 2 4 4 1 1 1 1 ...
##  $ NEWRACE2  : Factor w/ 7 levels "1","2","3","4",..: 1 7 7 1 5 1 1 2 1 1 ...
```

```
#find out how many complete cases we have
complete <- complete.cases(cc)
#get the percent missing
((nrow(cc) - sum(complete))/nrow(cc)) * 100
```

```
## [1] 0.3472385
```

Imputation

With the newly introduced NA's, our data has .35% missing data. We could probably drop the incomplete observations entirely. But, chances are we'd have some respondents that would also refuse to answer some questions, and in practice we will have far fewer observations to work with. So instead of dropping the observations we'll use the missForest package to impute the data for the risk variables. There was 14.6% out of box error in the imputation process. Hopefully, this will not negatively impact our predictions.

```
set.seed(13)
cc.imp <- missForest(cc[, -1]) #impute with everything but our response variable
```

```
##   missForest iteration 1 in progress...done!
##   missForest iteration 2 in progress...done!
##   missForest iteration 3 in progress...done!
##   missForest iteration 4 in progress...done!
##   missForest iteration 5 in progress...done!
```

```
cc.imp$OOBerror
```

```
##       PFC
## 0.1457113
```

```
#get the imputed values
ximp <- cc.imp$ximp


#add the imputed info back to the dataframe
cc$RSKCIGPKD <- ximp$RSKCIGPKD
cc$RSKMRJMON <- ximp$RSKMRJMON
cc$RSKMRJWK <- ximp$RSKMRJWK
cc$RSKLSDTRY <- ximp$RSKLSDTRY
cc$RSKLSDWK <- ximp$RSKLSDWK
cc$RSKHERTRY <- ximp$RSKHERTRY
cc$RSKHERWK <- ximp$RSKHERWK
cc$RSKCOCMON <- ximp$RSKCOCMON
cc$RSKCOCWK <- ximp$RSKCOCWK
```

```
cc$RSKBNGDLY <- ximp$RSKBNGDLY
cc$RSKBNGWK <- ximp$RSKBNGWK

#clear up our environment
rm(list=setdiff(ls(), "cc"))
```

Data Exploration

We can visually explore relationships between the predictor variables and our response variable. All of our variables are factors, and all except the response are coded with numbers. For ease of plotting, the numeric levels will be converted to more descriptive text labels in plots. Additionally, all plots will be done separately by the levels in the response variable ("No" and "Yes" for is and is not an opioid abuser).

Separating the plots by the response variable allows the reader to compare proportions within each predictor on the same scale. For example, if 50% of non-abusers are female and 50% of abusers are female, those are equal proportions. However, since only 6% of our observations are abusers, seeing that equality (or difference, as the case may be), when it is 50% of 6% is challenging. Separating by response makes it easier to see those results.

```
# for ease of output, give pretty labels to all the factors
temp <- cc
risk <- c('None', 'Slight', "Don't Know",  'Moderate', 'Great' )
rlevels <- c(-2,-1,0,1,2)
temp$RSKCIGPKD <- ordered(cc$RSKCIGPKD, levels=rlevels, labels=risk)
temp$RSKMRJMON <- ordered(cc$RSKMRJMON, levels=rlevels,labels=risk)
temp$RSKMRJWK <- ordered(cc$RSKMRJWK, levels=rlevels, labels=risk)
temp$RSKLSDTRY <- ordered(cc$RSKLSDTRY, levels=rlevels, labels=risk)
temp$RSKLSDWK <- ordered(cc$RSKLSDWK, levels=rlevels, labels=risk)
temp$RSKHERTRY <- ordered(cc$RSKHERTRY, levels=rlevels, labels=risk)
temp$RSKHERWK <- ordered(cc$RSKHERWK, levels=rlevels, labels=risk)
temp$RSKCOCMON <- ordered(cc$RSKCOCMON, levels=rlevels, labels=risk)
temp$RSKCOCWK <- ordered(cc$RSKCOCWK, levels=rlevels, labels=risk)
temp$RSKBNGDLY <- ordered(cc$RSKBNGDLY, levels=rlevels, labels=risk)
temp$RSKBNGWK <- ordered(cc$RSKBNGWK, levels=rlevels, labels=risk)


#ordered factors
temp$CATAG6 <- ordered(cc$CATAG6, labels=c('18-25', '26-34', '35-49', '50-64', '65+' ), levels=c(2,3,4,5,6))
temp$EDUHIGHCAT <- ordered(cc$EDUHIGHCAT, labels=c('Less than High School', 'High School Grad', 'Some College', 'College Grad'))
temp$HEALTH2 <- ordered(cc$HEALTH2, labels=c('Excellent', 'Very Good', 'Good', 'Fair/Poor'))

#nominal factors
temp$IRMARIT <- factor(cc$IRMARIT, labels=c('Married', 'Widowed', 'Divorced/Sep', 'Never Married'))
temp$IRSEX <- factor(cc$IRSEX, labels=c('Male', 'Female'))
temp$IRWRKSTAT <- factor(cc$IRWRKSTAT, labels=c('Full Time', 'Part Time', 'Unemp.', 'Other'))
temp$NEWRACE2 <- factor(cc$NEWRACE2, labels=c('NonHisp White', 'NonHisp Black/Afr Am', 'NonHisp Native Am/AK Native', 'NonHisp Navite HI/Other Pac Isl', 'NonHisp Asian', 'NonHisp more than 1 race', 'Hispanic'))

str(temp)

#set up a function to do easy side by side plots
doSideBySidePlot <- function(column, name, ylim=1, risk=F){
  scaleName = name
  if(risk==T){
    scaleName = 'Risk'
  }
  no <- ggplot(temp %>% dplyr::filter(temp$isUser == "No"), aes(isUser, (..count..)/sum(..count..))) + geom_bar(aes(fill = get(column)), position = "dodge") + ggtitle(paste(name, 'Not an Abuser', sep=": ")) +
  ylab("Percent") +
  fill_palette("jco") +
  xlab("") +
  guides(fill=guide_legend(title=scaleName)) +
  scale_y_continuous(limits = c(0, ylim))


yes <- ggplot(temp %>% dplyr::filter(temp$isUser == "Yes"), aes(isUser, (..count..)/sum(..count..))) + geom_bar(aes(fill = get(column)), position = "dodge") + ggtitle(paste(name, 'Abuser', sep=": ")) +
   ylab("Percent")+
  fill_palette("jco")+
  xlab("") +
  scale_y_continuous(limits = c(0, ylim))

return(grid_arrange_shared_legend(no, yes, ncol=2))


}
```

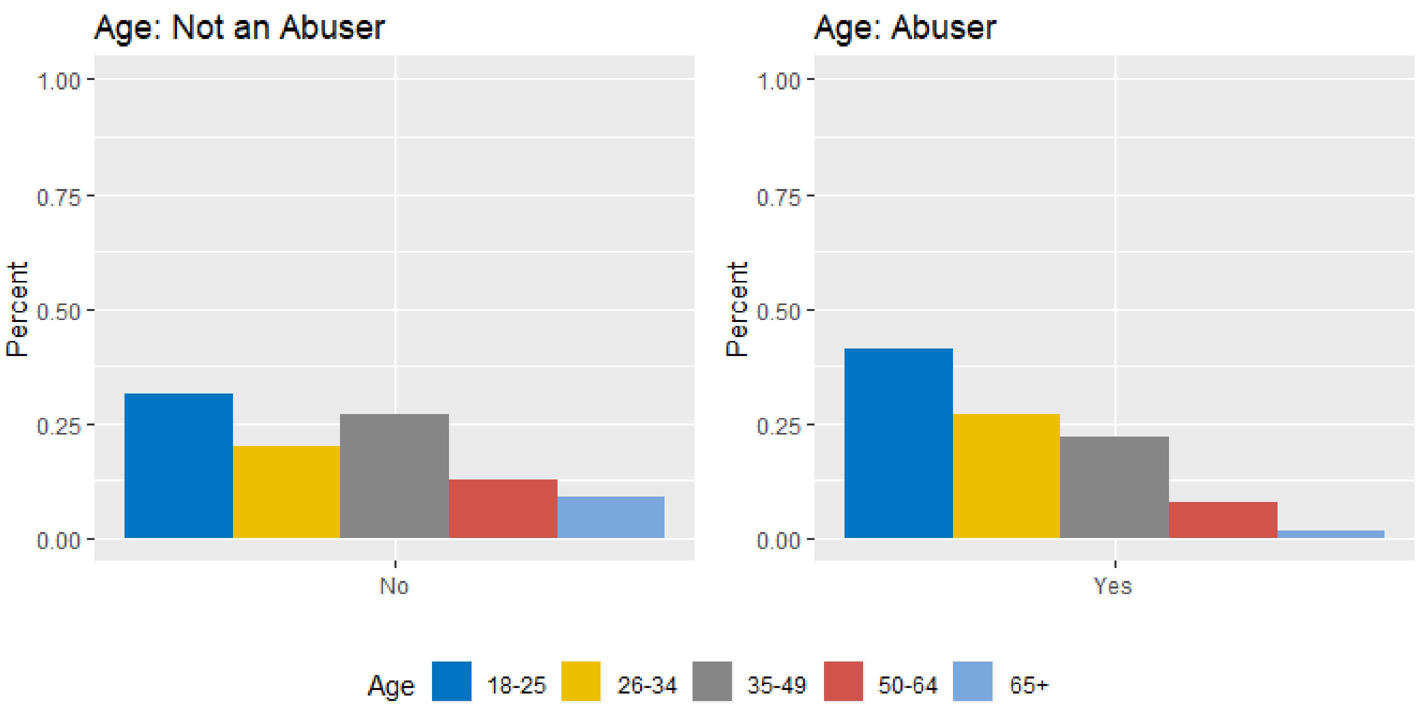
Demographic Information

Age

The age variable suggests that people under 35 have a higher tendency to be abusers, while people over 35 have a lower tendency to be abusers. This is most apparent in the youngest and oldest age groups (18-25 & 65+).

```
doSideBySidePlot('CATAG6', 'Age', 1)
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.1
```

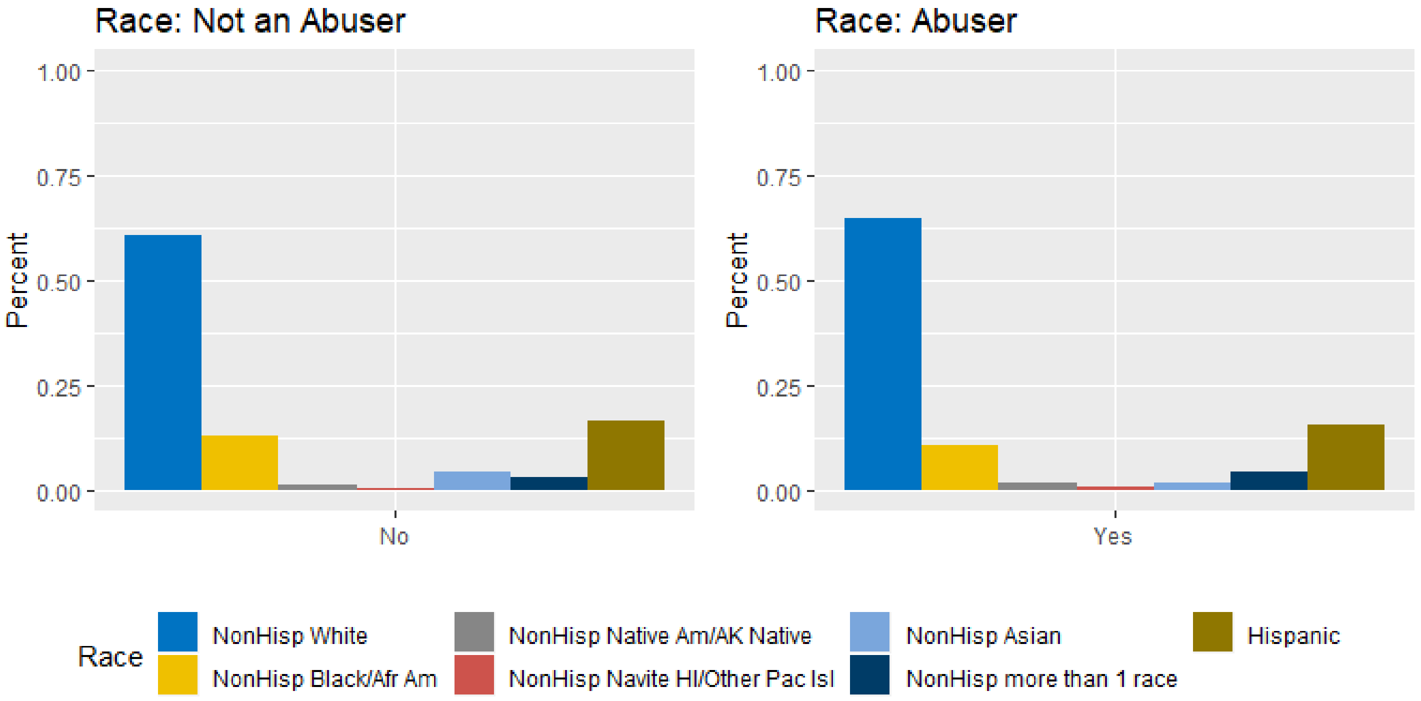


Race

Race is a delicate subject when illegal behavior is involved. Ethically, one must be cautious when using it as a predictor. However, in consultation with the client, it was decided that race could be ethically used in this case, because the results of the analysis are meant to be used in aggregate and for prevention work.

When looking at the proportions in each race category, non-Hispanic whites have the largest percentage difference between users and nonusers (with the higher percentage being users). However, all differences by race are subtle.

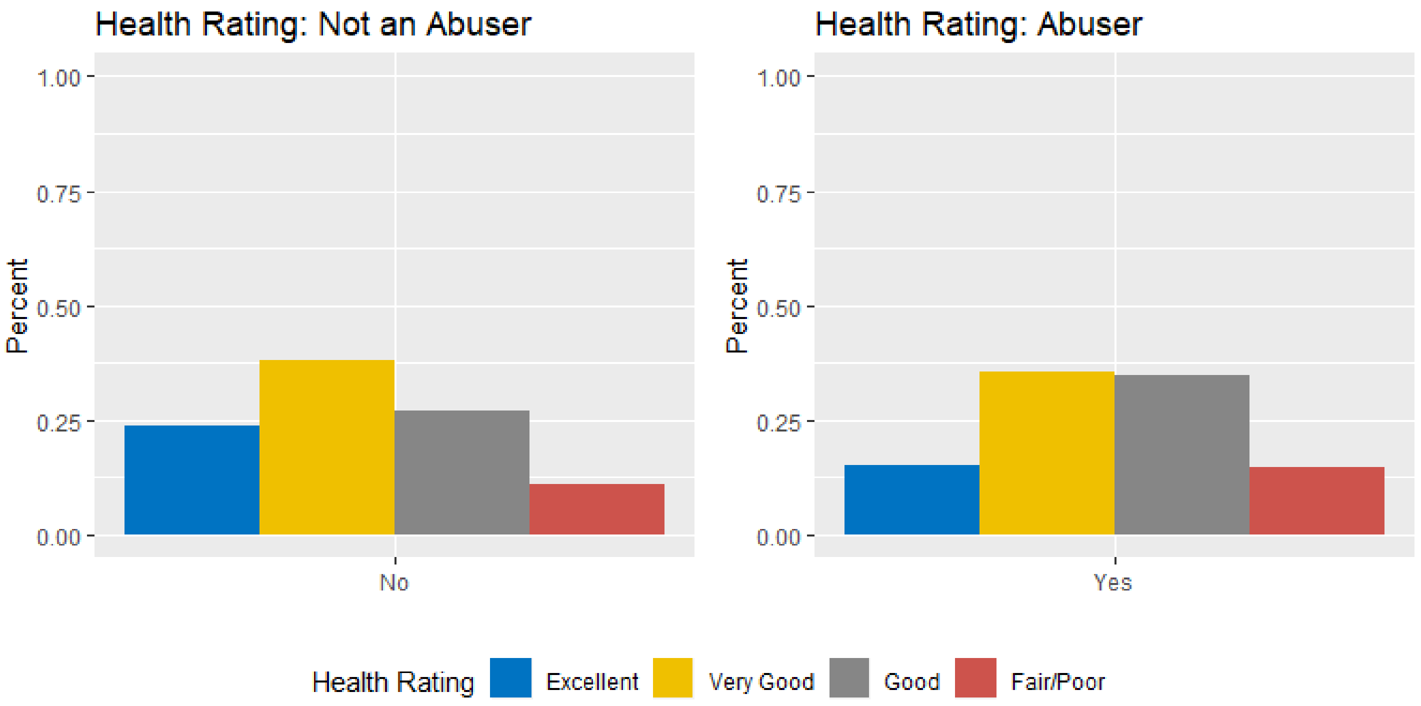
```
doSideBySidePlot('NEWRACE2', 'Race', 1)
```



Health

The health indicator was one of interest to the client. She claimed that perception of health is one of the leading indicators of mortality rate. (I have not sought out verification of that claim.) In our dataset, it appears that non-abusers are more likely to rank their health as excellent or very good, while abusers are more likely to rank their health as good or fair/poor. Note that this scale is interesting in that there are 3 positive responses and only one negative response, and the scale of this question is also transposed (positive value = low number).

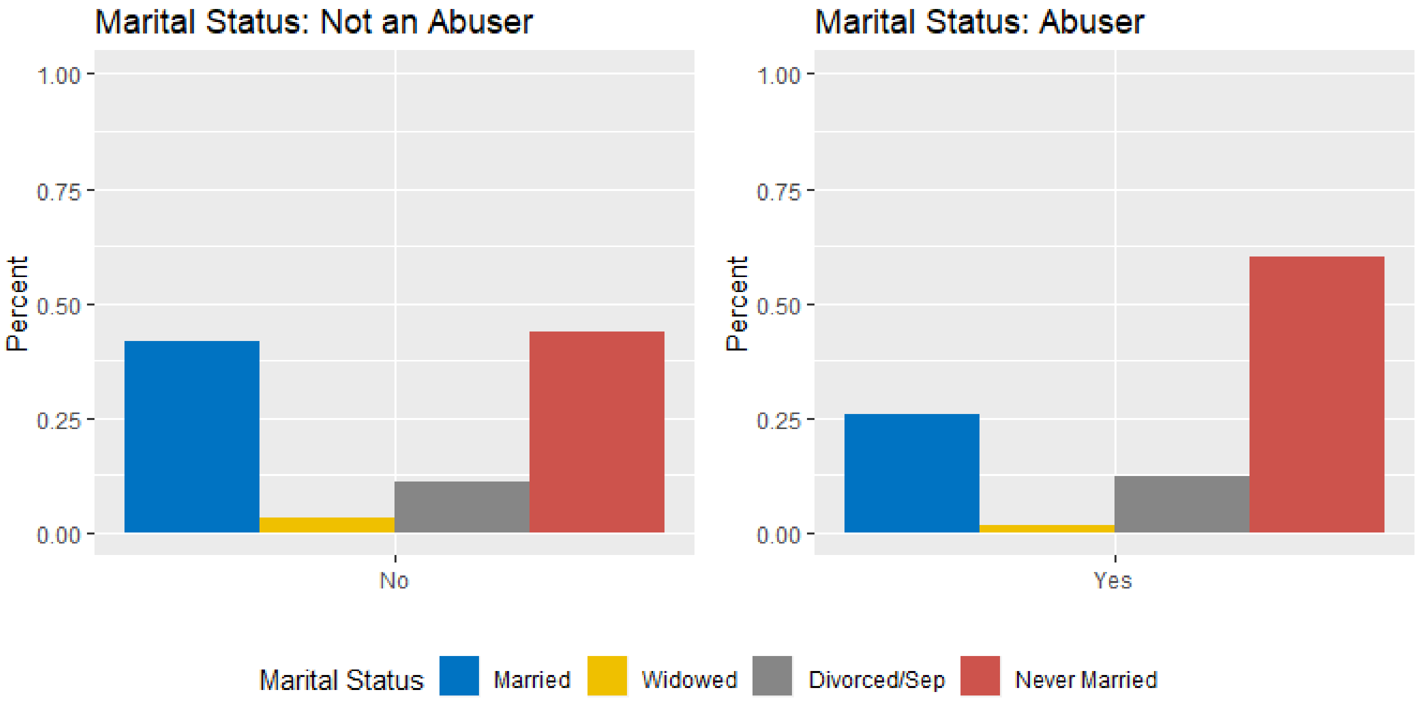
```
doSideBySidePlot('HEALTH2', 'Health Rating', 1)
```



Marital Status

In the marital status variable, a much larger percentage of abusers have never been married (and a much lower percentage are currently married). The percentages of widowed and divorced/separated are roughly equal.

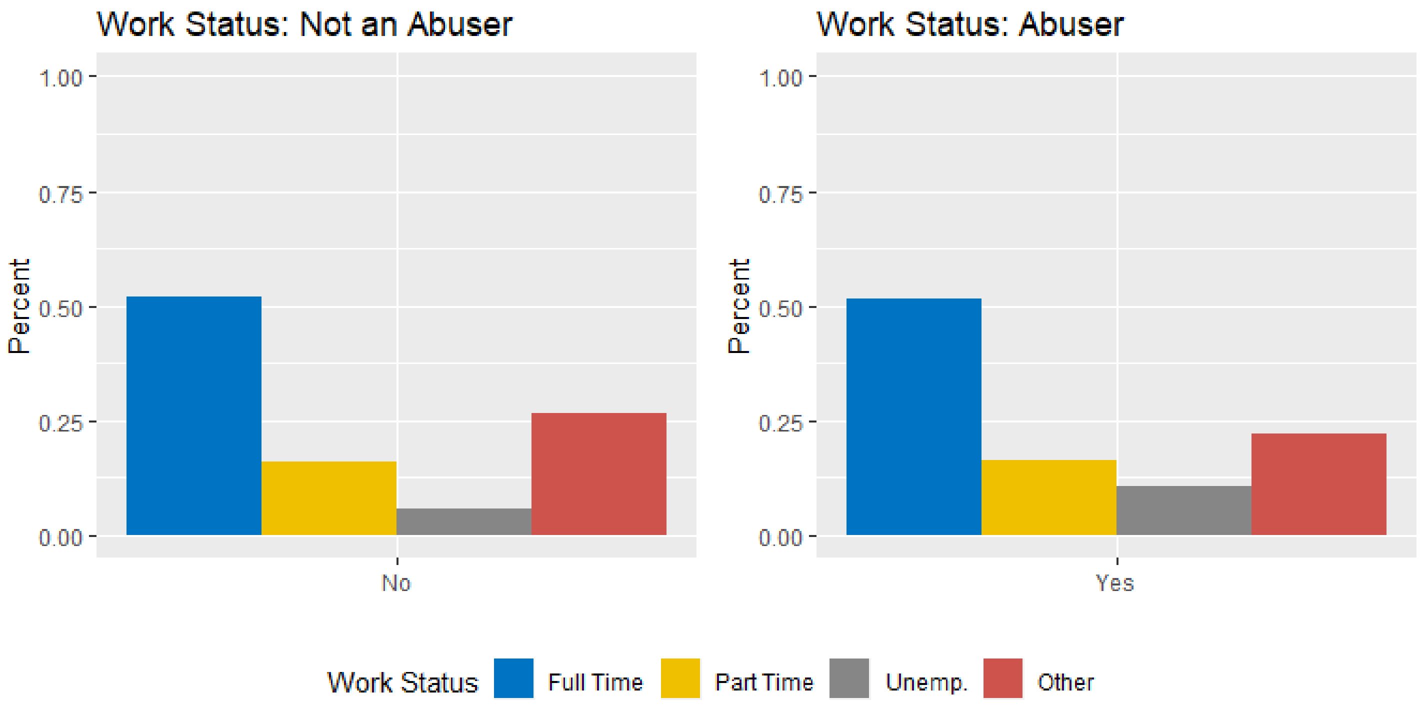
```
doSideBySidePlot('IRMARIT', 'Marital Status', 1)
```



Work Status

The work status variable shows nearly even proportions throughout. There is a marginally higher percentage of unemployed individuals who are abusers, and a marginally smaller percentage of "other" individuals. The “other” category would include people who are retired or who are on disability and not in the workforce. The slightly lower percent in other here might correlate with the lower percent of people over 65 who are abusers that we saw earlier.

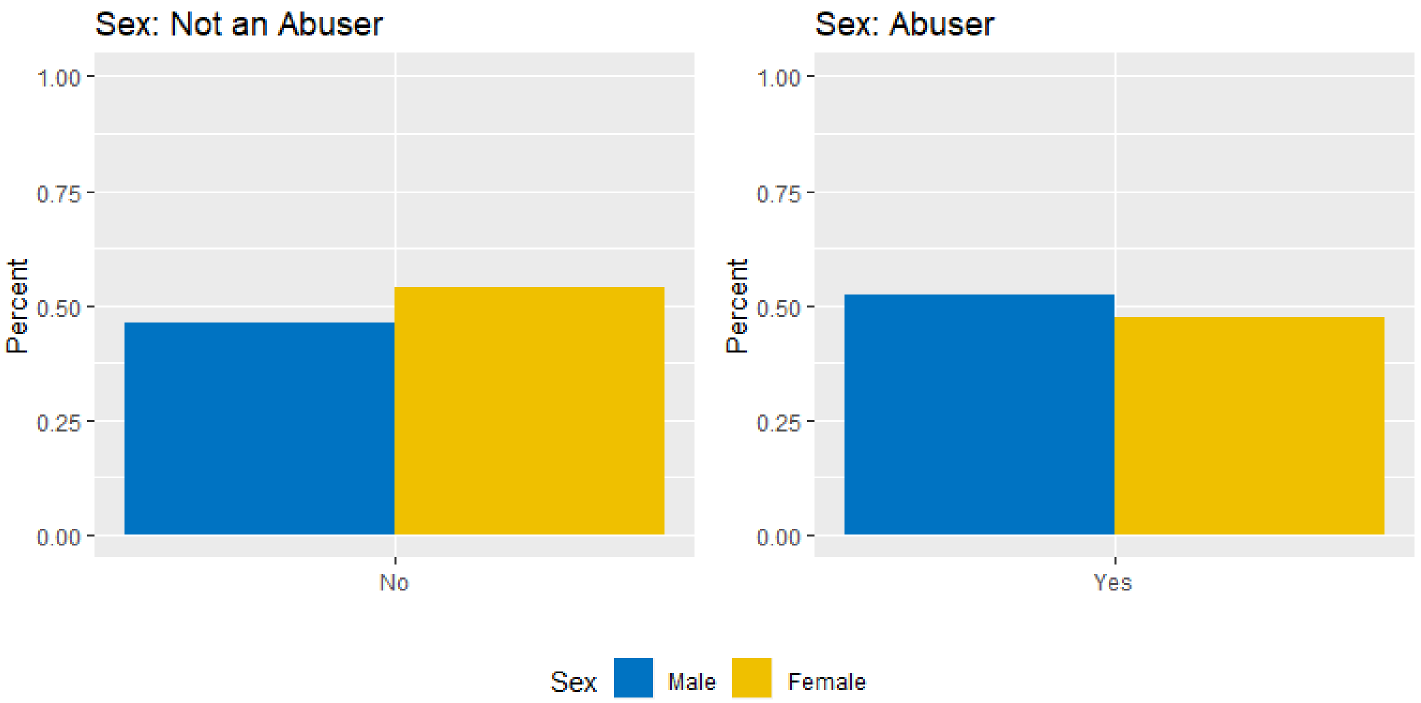
```
doSideBySidePlot('IRWRKSTAT', 'Work Status', 1)
```



Sex

Individuals were forced to choose one of 2 sexes - male or female. In 2016, the U.S. population breakdown was 51% female and 49% male. If gender had no impact on opioid use, you’d expect to see those breakdowns here. Instead what we see here is that among abusers, those proportions are approximately reversed.

```
doSideBySidePlot('IRSEX', 'Sex', 1)
```



Risks Factors

For each of the risk factor questions, respondents were asked to assess how much they think people risk harming themselves physically and in other ways when they do each of the following activities.

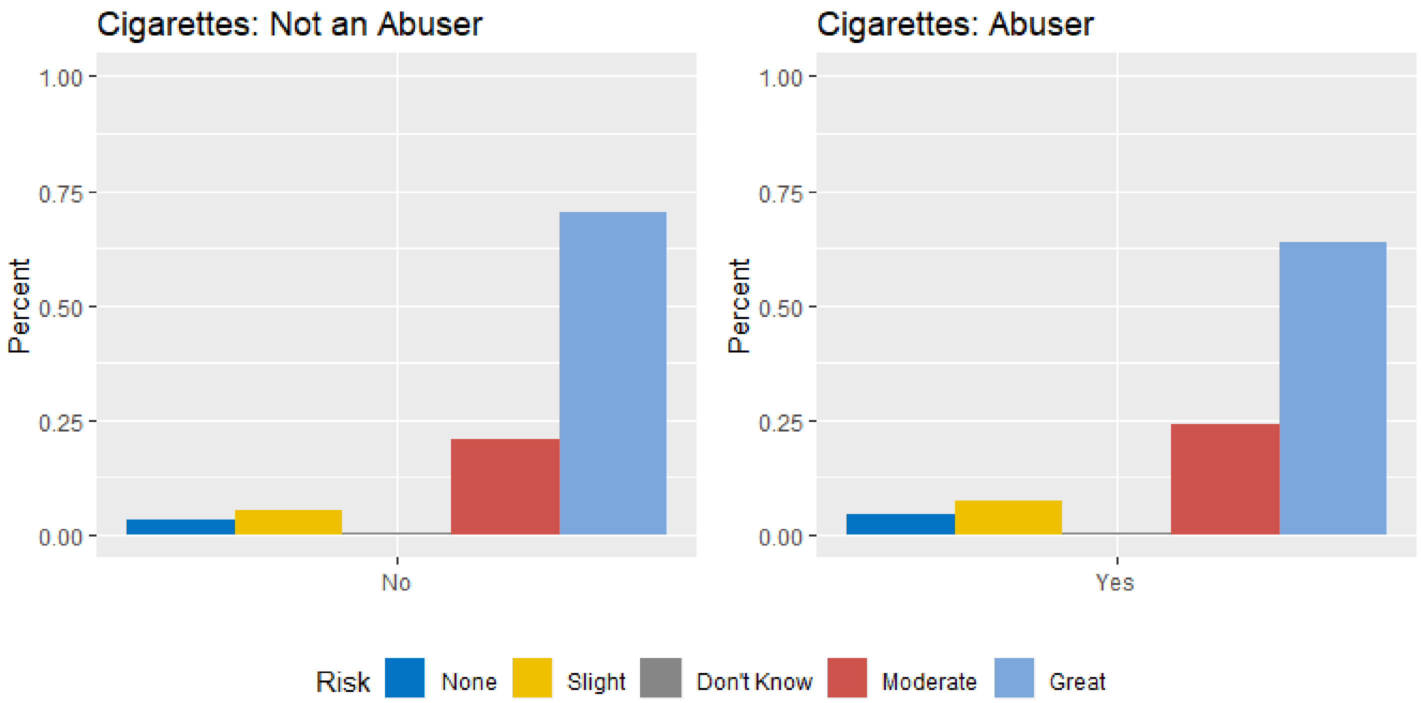
The risk factors are an ordered numeric scale going from 0 (no risk) to 5 (great risk), with our neutral "Don't Know" in the middle at 3. Again, we can visualize these with bar charts.

Respondents were asked about the risk of two legal substances: drinking and cigarette smoking, and 4 illegal substances: marijuana, LSD, heroin, and cocaine.

Smoking

For smoking, respondents were asked to assess the risk of smoking a pack or more of cigarettes a day. The two classes of respondents gave similar assessments, with a slightly higher percentage of non-abusers deeming it "great" risk.

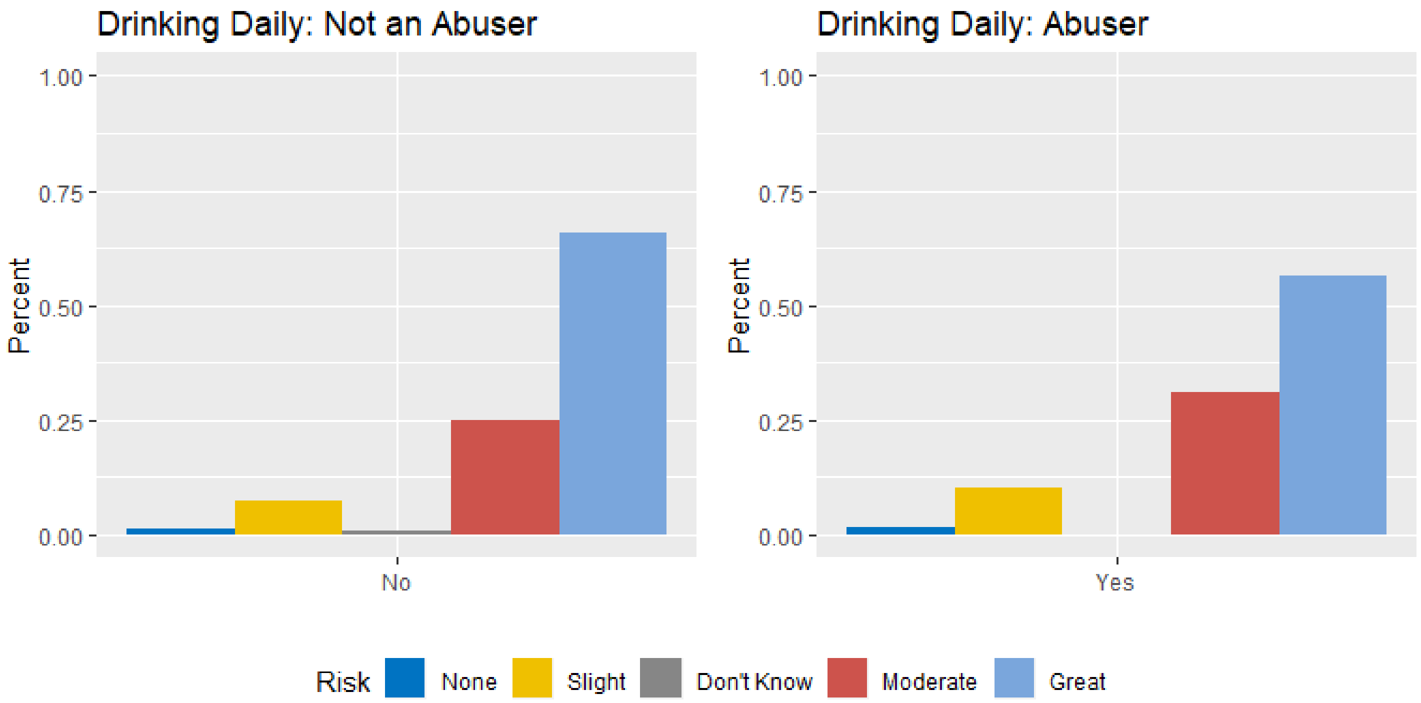
```
doSideBySidePlot('RSKCIGPKD', 'Cigarettes', 1, risk = T)
```



Binge Drinking Daily

Binge drinking was defined as drinking four or five drinks in one sitting. Respondents were asked about the risk of binge drinking daily. Non-abusers had a slightly higher tendency to rate this as great risk, while abusers had a slightly higher tendency to rate this as moderate risk.

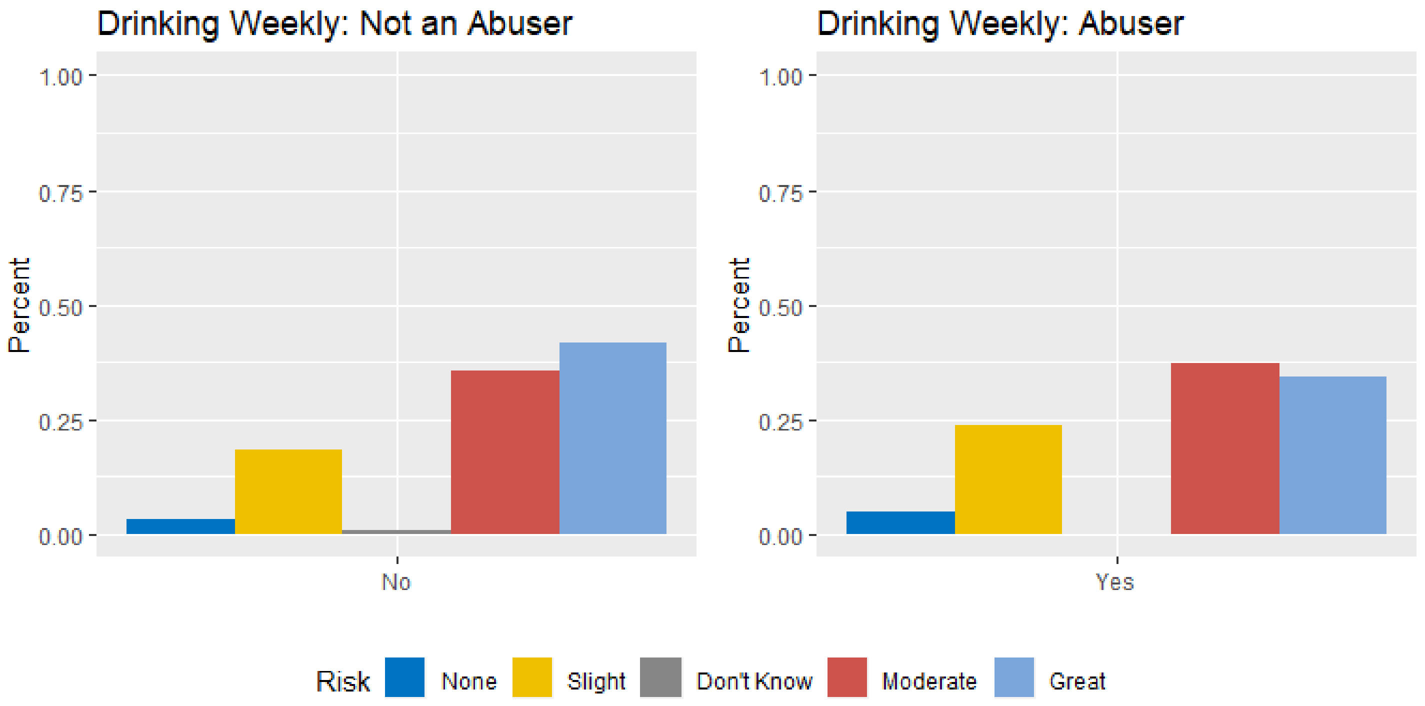
```
doSideBySidePlot('RSKBNGDLY', 'Drinking Daily', 1, risk=T)
```



####Drinking Weekly

Respondents were also asked about binge drinking weekly. Both groups viewed this as less risky behavior than drinking daily. Again, the abusers tended to view this as more moderate while the non-abusers tended to view this as greater risk.

```
doSideBySidePlot('RSKBNGWK', 'Drinking Weekly', 1, risk=T)
```

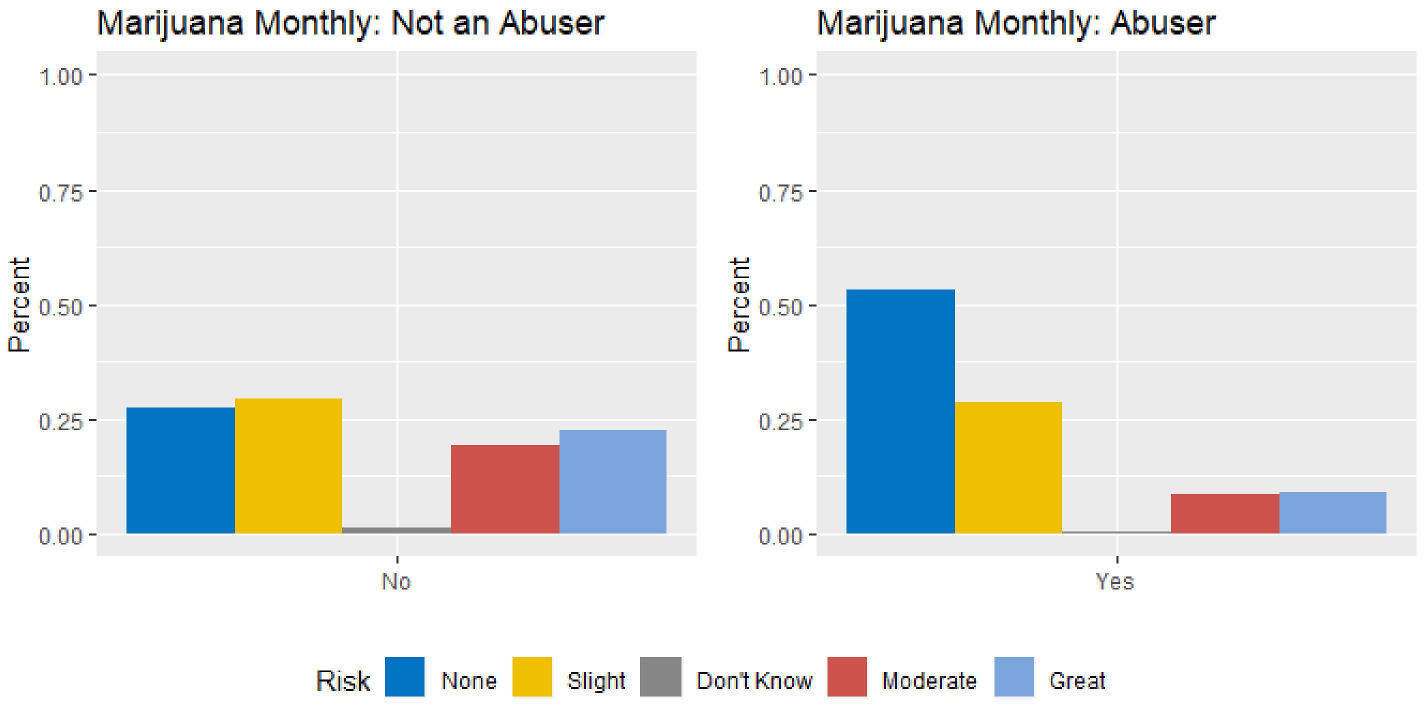

Drinking Weekly: Not an Abuser

Drinking Weekly: Abuser

Marijuana Monthly

Respondents were asked about the risk of using marijuana monthly. This is our first variable that shows stark differences between abusers and non-abusers, with abusers much more likely to select no risk than abusers.

```
doSideBySidePlot('RSKMRJMON', 'Marijuana Monthly', 1, risk=T)
```

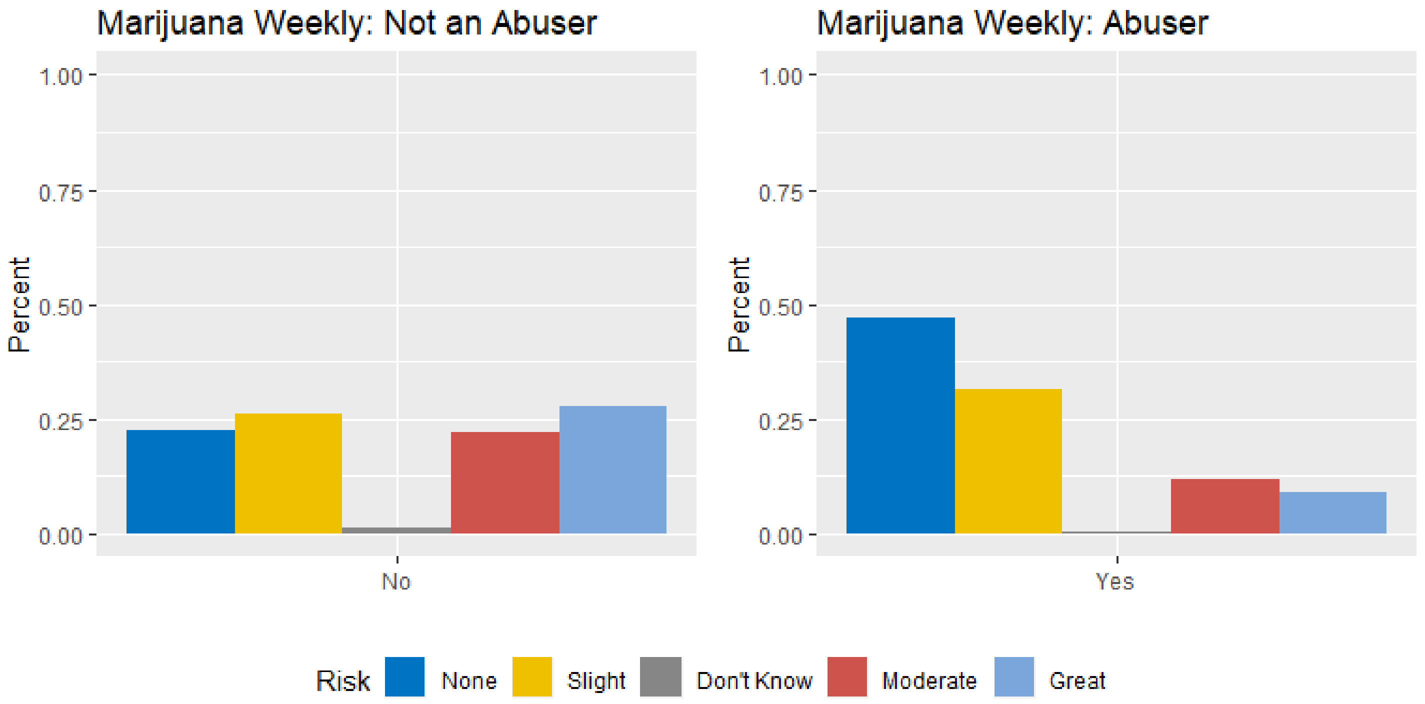

Marijuana Monthly: Not an Abuser

Marijuana Monthly: Abuser

Marijuana Weekly

Respondents were also asked about the risk of using marijuana weekly. Again, there is a noticeable difference between abusers and non-abusers, with abusers much more likely to view this as a no or slight risk activity.

```
doSideBySidePlot('RSKMRJWK', 'Marijuana Weekly', 1, risk=T)
```

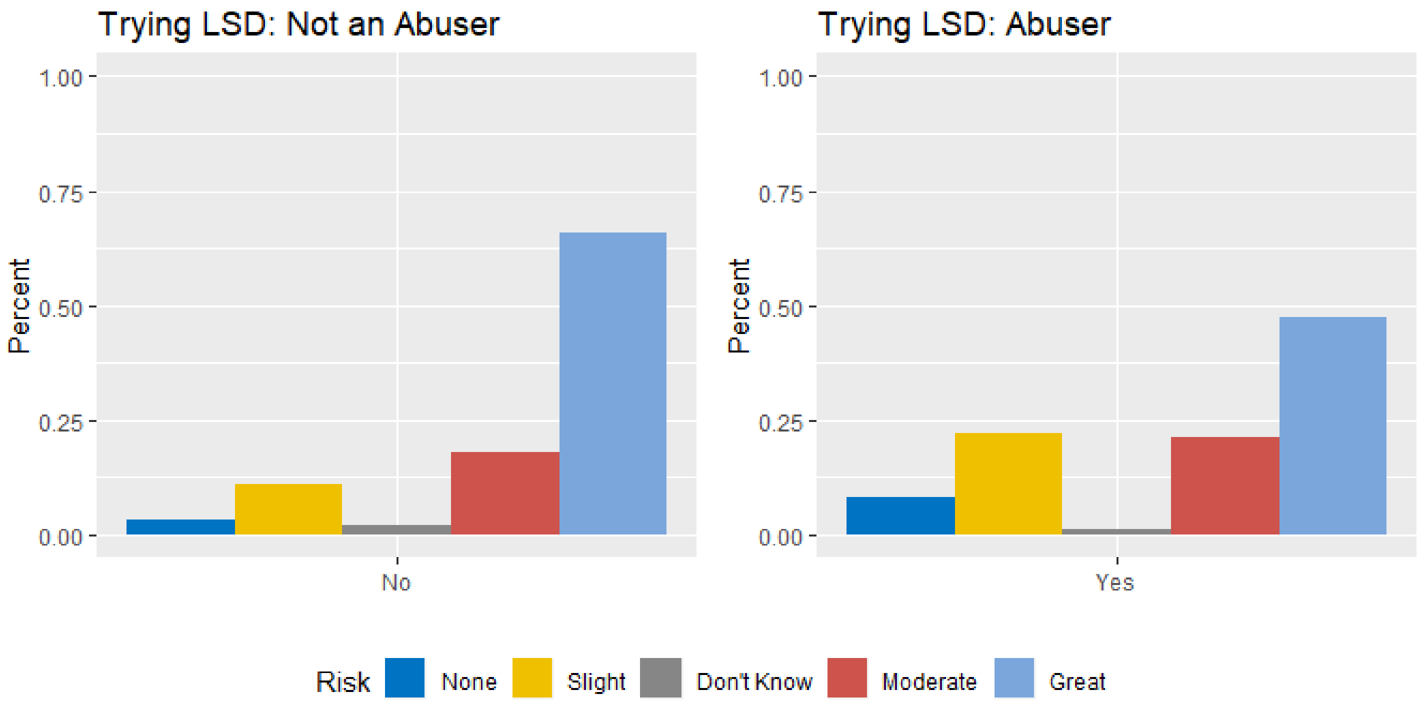


```
#do zoomed in versions of both marijuana plots for final report
mjwk <- doSideBySidePlot('RSKMRJWK', 'Marijuana Weekly', .6, risk=T)
ggsave('mjweekly.png', mjwk)
mjmon <- doSideBySidePlot('RSKMRJMON', 'Marijuana Monthly', .6, risk=T)
ggsave('mjmonthly.png', mjmon)

#get actual percentatages for monthly use
prop.table(table(cc$RSKMRJMON[which(cc$isUser == 'No')]))
prop.table(table(cc$RSKMRJMON[which(cc$isUser == 'Yes')]))

#get actual percentatages for weekly use
prop.table(table(cc$RSKMRJWK[which(cc$isUser == 'No')]))
prop.table(table(cc$RSKMRJWK[which(cc$isUser == 'Yes')]))
```

Trying LSD

Respondents were asked about the risk of trying LSD. Both abusers and non-abusers considered this a riskier behavior than marijuana, with non-abusers tending to view it as riskier than abusers.

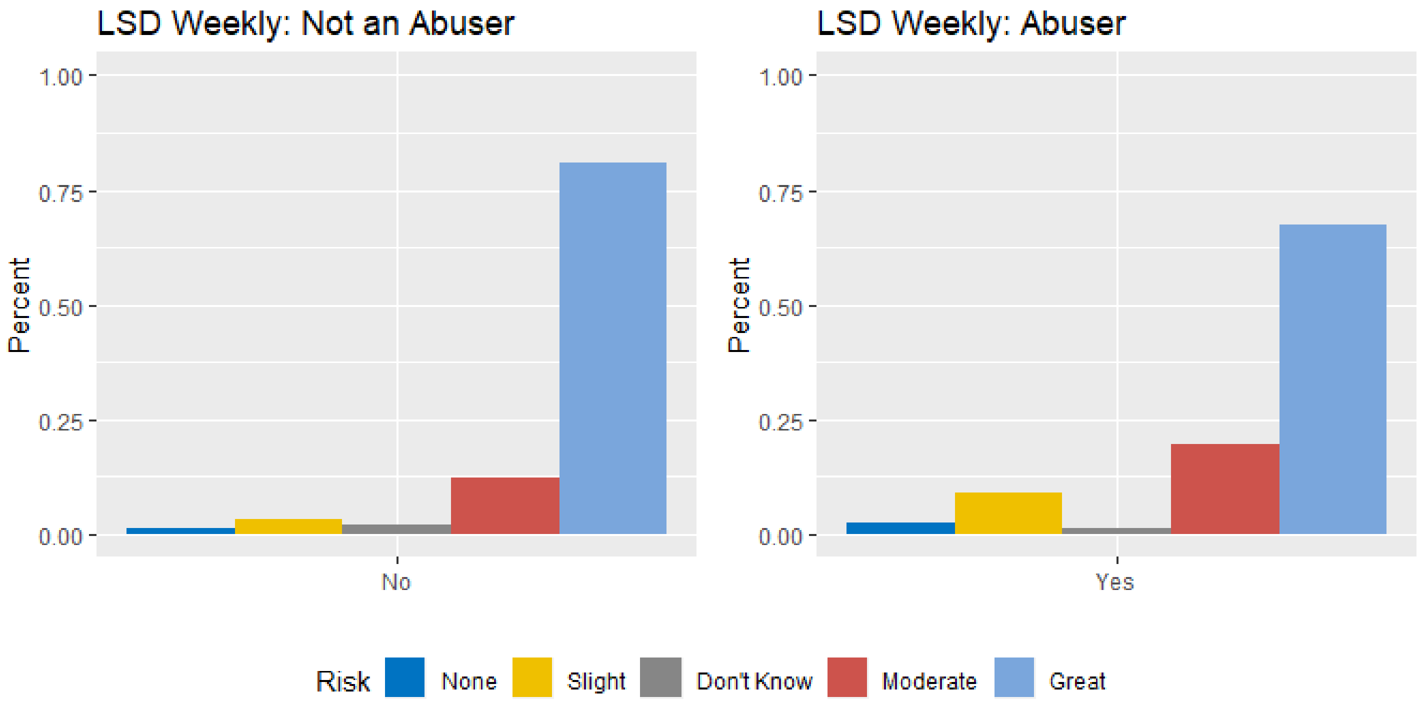
```
doSideBySidePlot('RSKLSDTRY', 'Trying LSD', 1, risk=T)
```



LSD Weekly

Respondents were also asked about using LSD weekly. Weekly use was deemed riskier by both groups, with about the same difference between the groups as seen in trying it.

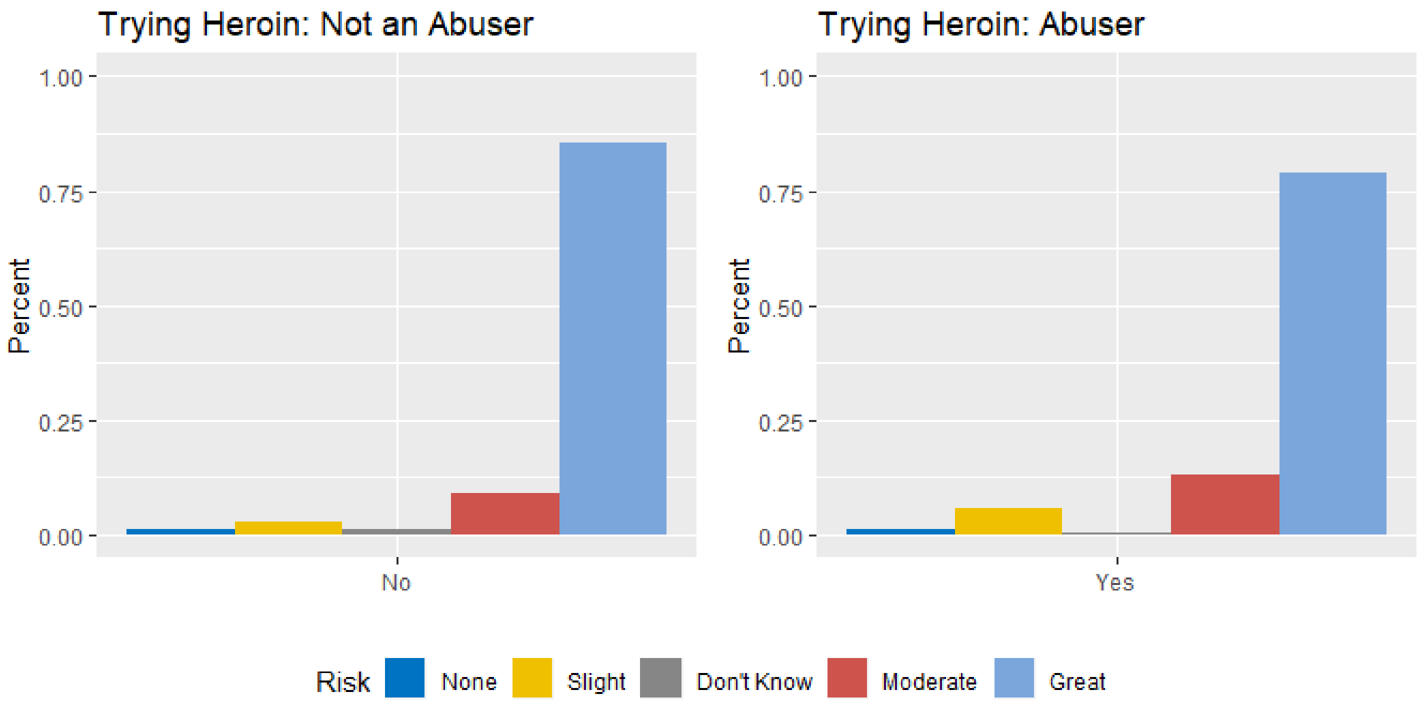
```
doSideBySidePlot('RSKLSDWK', 'LSD Weekly', 1, risk=T)
```



Trying Heroin

Trying heroin is seen as very risky by both groups. Again, the abuser group sees it as slightly less risky than the non-abuser group.

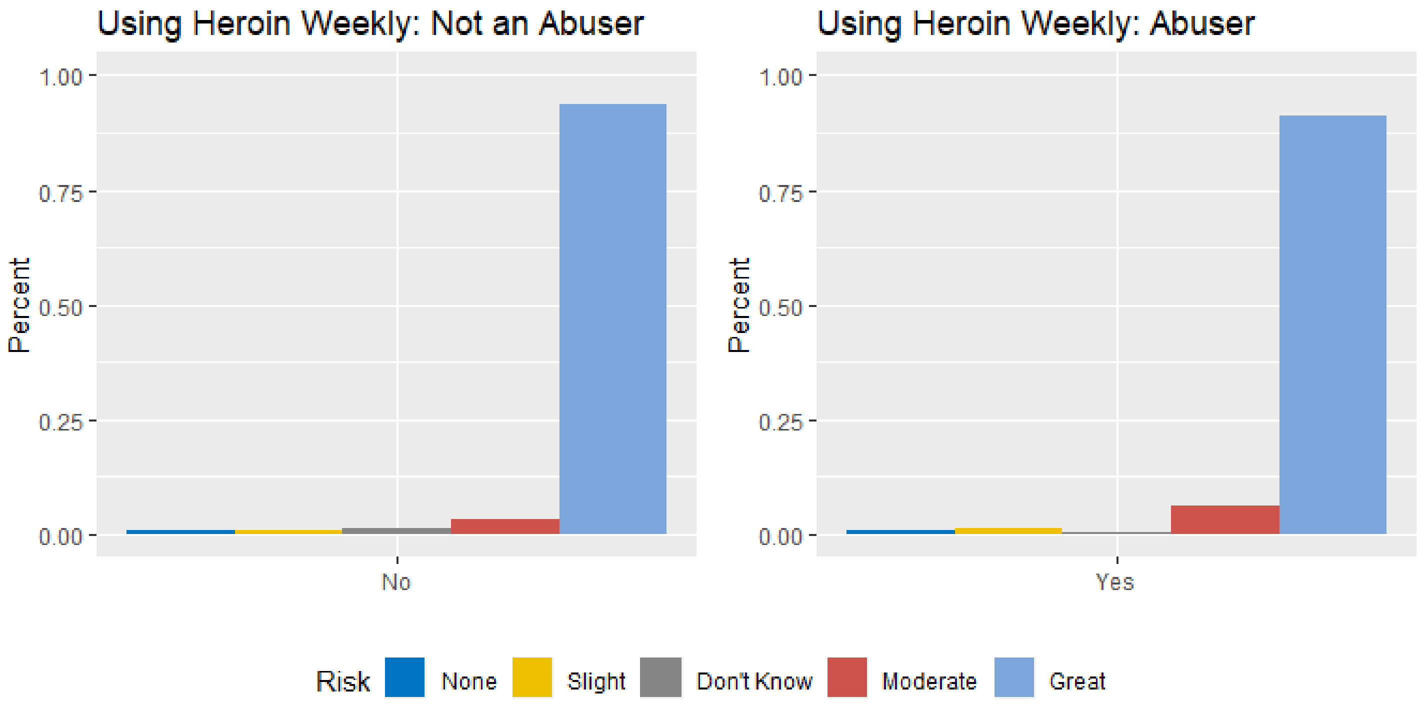
```
doSideBySidePlot('RSKHERTRY', 'Trying Heroin', 1, risk=T)
```



Heroin Weekly

Both abusers and non-abusers view using heroin weekly as a great risk. There is only marginal difference between the two groups where this substance is concerned.

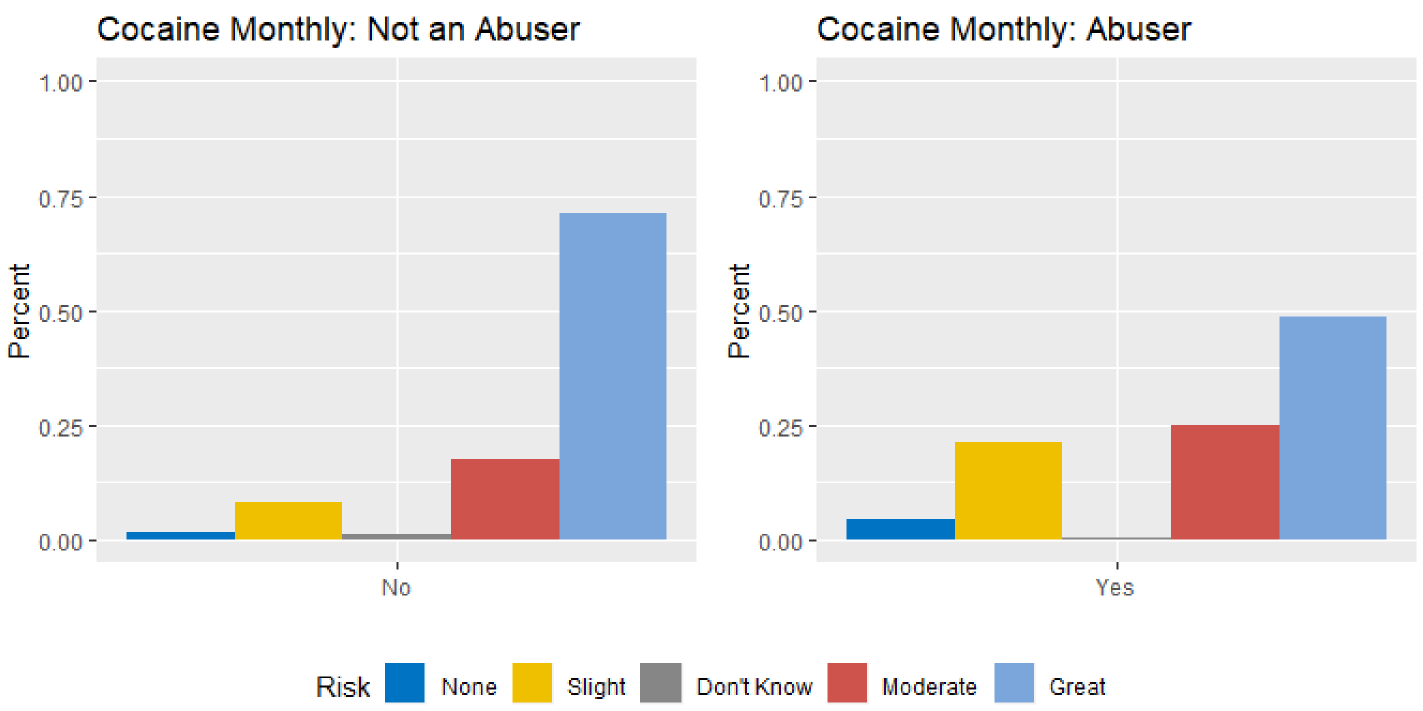
```
doSideBySidePlot('RSKHERWK', 'Using Heroin Weekly', 1, risk=T)
```



Cocaine Monthly

Using cocaine monthly is another risk factor that shows fairly significant differences, with non-abusers viewing this practice as much riskier than abusers.

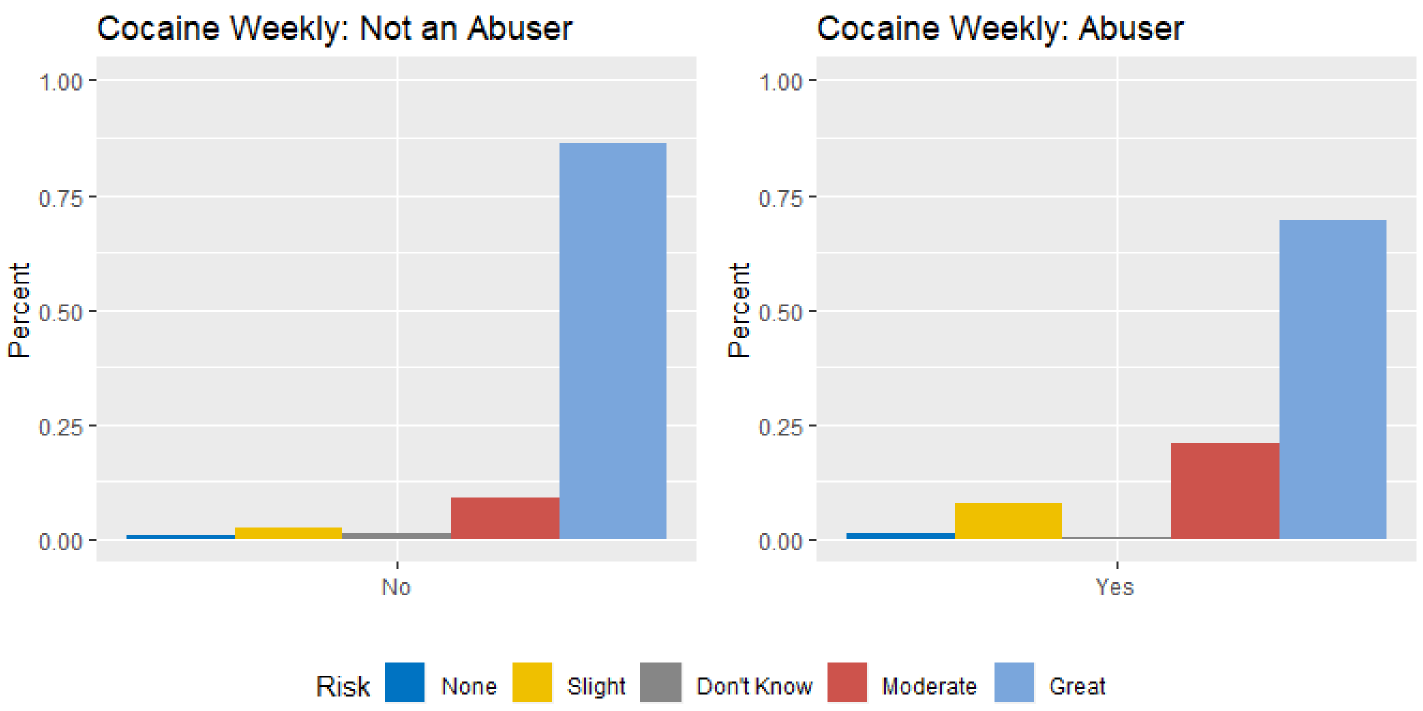
```
doSideBySidePlot('RSKCOCMON', 'Cocaine Monthly', 1, risk=T)
```



Cocaine Weekly

Using cocaine weekly is seen as a great risk more often by non-abusers than abusers. Abusers tend to rank weekly cocaine use as moderate or slight risk more often than non-abusers.

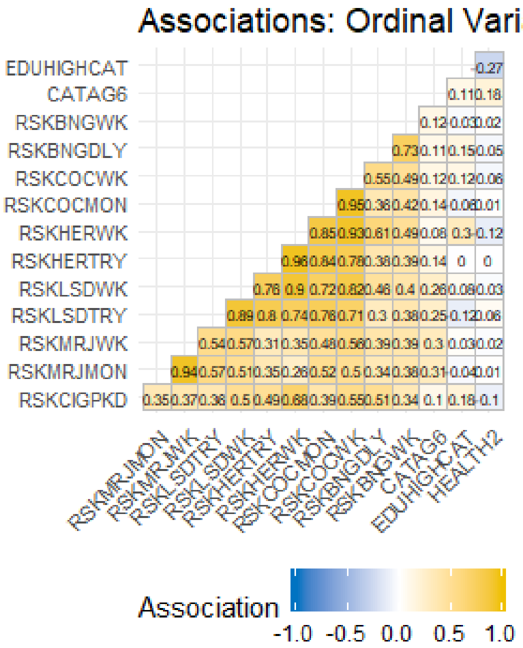
```
doSideBySidePlot('RSKCOCWK', 'Cocaine Weekly', 1, risk=T)
```



Measures of Association

Another way to assess categorical variables is to use measures of association. Goodman and Kruskal's gamma test measures the strength of association between 2 ordinal variables in a symmetrical way (one variable is not dependent on the other). Gamma will range from -1 to 1, with zero indicating no association and 1 or -1 indicating perfect positive or negative association (Mangiafico, 2016). We can look at the association of each of the risk factors to each other and to the ordinal demographic variables. From this plot we can see that all the risk variables are associated to some degree, with the lowest association between risk variables being .26. Some variables are strongly associated with each other, such as the risk of trying heroin and the risk of using heroin weekly. The demographic variables are less strongly associated. This suggests looking at models that could eliminate one of the variables from each high association pair or looking at models that can handle variables that might provide similar information, like random forests and penalized approaches.

```
##helper function to make an association matrix for
getKGMatrix <- function(cols){

  l <- length(cols)
  #make a matrix the same size as the columns passed in
  results <- matrix(, nrow = length(cols), ncol = length(cols))
  rownames(results) <- cols
  colnames(results) <- cols

  for (c in 1:l){
    #get this column and a vector of the other columns
    thisCol <- cols[c]
    others <- cols

    #loop through all the columns
    for(o in 1:length(others)){
      nextCol <- cols[o]
      tabla <- table(cc[[thisCol]], cc[[nextCol]])
      kg <- GoodmanKruskalGamma(tabla, conf.level=.95)
      results[[thisCol, nextCol]] <- kg['gamma']
      results[[nextCol, thisCol]] <- kg['gamma']
    }


  }
  return(results)
}

#make a vector of all the risk variables
ords <- c('RSKCIGPKD', 'RSKMRJMON', 'RSKMRJWK', 'RSKLSDTRY', 'RSKLSDWK', 'RSKHERTRY',
              'RSKHERWK', 'RSKCOCMON',
              'RSKCOCWK', 'RSKBNGDLY', 'RSKBNGWK', 'CATAG6', 'EDUHIGHCAT', 'HEALTH2')

riskMatrix <- getKGMatrix(ords)


#plot this
plot1 <- ggcorrplot(riskMatrix, title = "Associations: Ordinal Variables", lab=TRUE,   type = "lower", lab_size = 2, hc.order
= FALSE, tl.cex = 8, colors=c("#0073C2FF", "#EFC000FF", "#FFFFFFFF")) + theme(plot.margin = unit(c(.5,.5,.5,.5), "cm
"), legend.position="bottom")  +
  scale_fill_gradient2(
      low = '#0073C2FF', high = '#EFC000FF', mid = '#FFFFFFFF',
      midpoint = 0, limit = c(-1,1), space = "Lab",
      name = "Association"
    )

## Scale for 'fill' is already present. Adding another scale for 'fill',
## which will replace the existing scale.

plot1
```

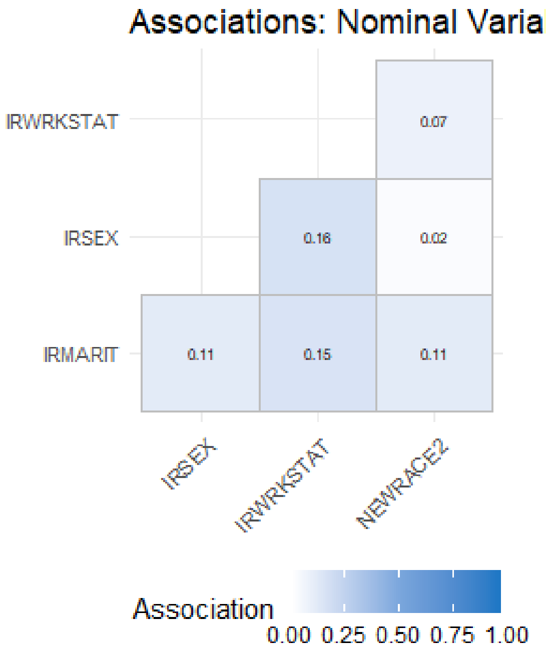


```
ggsave('RiskAssociations.png')

## Saving 8 x 4 in image
```

Cramer's V is a statistic that shows the strength of the association between two nominal variables. V varies from 0 to 1, with zero being no association and 1 being a strong association. We have just 4 nominal predictors, none of which are strongly associated with one another.

```
##helper function to make an association matrix for
getVMatrix <- function(cols){


  l <- length(cols)
  #make a matrix the same size as the columns passed in
  results <- matrix(, nrow = length(cols), ncol = length(cols))
  rownames(results) <- cols
  colnames(results) <- cols

  for (c in 1:l){
    #get this column and a vector of the other columns
    thisCol <- cols[c]
    others <- cols

    #loop through all the columns
    for(o in 1:length(others)){
      nextCol <- cols[o]
      tabla <- table(cc[[thisCol]], cc[[nextCol]])
      V <- cramerV(tabla, digits=2)
      results[[thisCol, nextCol]] <- V
      results[[nextCol, thisCol]] <- V
    }


  }
  return(results)
}
```

```
#make a vector of all the risk variables
noms <- c('IRMARIT', 'IRSEX', 'IRWRKSTAT', 'NEWRACE2')

nomMatrix <- getVMatrix(noms)


#plot this
plot2 <- ggcorrplot(nomMatrix, title = "Associations: Nominal Variables", lab=TRUE,   type = "lower", lab_size = 2, hc.order = FALSE, tl.cex = 8, colors=c("#0073C2FF", "#EFC000FF", "#FFFFFFFF")) + theme(plot.margin = unit(c(.5,.5,.5,.5), "cm"), legend.position="bottom")  +
  scale_fill_gradient2(
      low = '#FFFFFFFF', high = '#0073c299', mid = '#7AA6Dc99',
      midpoint = .5, limit = c(0,1), space = "Lab",
      name = "Association"
    )

## Scale for 'fill' is already present. Adding another scale for 'fill',
## which will replace the existing scale.

plot2
```



```
ggsave('NomAssociations.png')

## Saving 8 x 4 in image
```

Training and Validation Sets

This is a large dataset, with just under 43,000 rows of data. With a dataset this large, we can use the training and validation set approach as opposed to a double cross validation set approach.

```
#the caret library createdatapartition function makes better partitions than sampling yourself, because it does so from within factor levels
set.seed(13)
trainIndex <- createDataPartition(cc$isUser, p=.7, list=F)
cc.train <- cc[trainIndex, ]
cc.test <- cc[-trainIndex, ]
```

Processing power/time

This dataset is too large to reasonably crunch through on a regular basis. So, for development purposes, I’m going to make a smaller split of the data. This won’t ever be run when the project is knit. It will just be used for proof of concept.

```
set.seed(13)
trainIndex <- createDataPartition(cc.train$isUser, p=.1, list=F)
cc.train <- cc[trainIndex, ]

table(cc.train$isUser)
```

Imbalanced Data

The data from this survey is extremely imbalanced. Less than 6% of the observations are opioid abusers. Imbalanced data can be handled several different ways. Weiss, et. al. suggest that while there is no hard and fast rule between under and over sampling techniques versus cost-sensitive learning techniques, cost sensitive techniques tend to produce better results when the dataset is large. A mixture of approaches will be used to address the imbalanced data, including some cost sensitive techniques (neural network and penalized logistic regression), some stratified in-algorithm sampling (random forest), and some techniques that use a threshold cutoff within the algorithm itself for training (random forest). In addition, model selection and assessment will both use a weighted threshold. Final model selection will be based on a weighted threshold accuracy score (using Youden’s approach), that sets a minimum sensitivity and specificity of .6. This ensures a model that is not biased towards picking the majority class and honors the client’s request that model have the both the highest sensitivity and specificity possible. Other measures were tried as well, including F-score and Kappa. (Note that seeking the highest value for both sensitivity and specificity for a model that does not have great predictive power means that we end up with mediocre sensitivity and specificity. It might be more optimal to focus on one or the other, but these decisions need to be made in consultation with the client, and this client chose a more balanced approach.)

```
getBestWeightedThreshold <- function(probYes, reference, type="auc", printResults=F){
  #get the prevalence of the 'Yes' values
  prev <- prop.table(table(reference))[2]

  #get a roc object
    fRoc <- roc(response=reference,
                predictor=probYes,
                auc=T,
                levels=c('No', 'Yes') #controls, cases
                )

    #get the area under curve
    AUC <- fRoc$auc

        #set up a dataframe to track weights and results
    weightResults <- data.frame('weight'=seq(1:30), 'AUC' = rep(NA, 30),  'threshold' = rep(NA, 30), 'accuracy'= rep(NA, 3
0), 'specificity'= rep(NA, 30), 'sensitivity'= rep(NA, 30), 'overall' = rep(NA, 30))
    for(w in 1:30){
      #get the best threshold & stats
        best<- coords(fRoc, x="best",
                      best.method="youden",
                      ret = c("threshold", "accuracy",  "specificity", "sensitivity"),
                      best.weights=c(w, .06) #relative cost of a fn vs. fp, prevalence, which we know is at .06 in our data
                      )
    weightResults$weight[w] <- w
    weightResults$AUC[w] <- AUC
    weightResults$threshold[w] <- best[1]
    weightResults$accuracy[w] <- best[2]
    weightResults$specificity[w] <- best[3]
    weightResults$sensitivity[w] <- best[4]
    weightResults$overall[w] <- best[2] #highest accuracy given the minimum thresholds

  }

  if(printResults == T){
   print(weightResults)
  }

  #We want a minimum specificity and sensitivity of .6
  meetsMin <- weightResults[which(weightResults$specificity > .6 & weightResults$sensitivity > .6), ]


  if(type=="accuracy"){
   #we want highest accuracy score, given the minimum thresholds
   bestWeights <- meetsMin[order(-meetsMin$overall), ]
  }
  if(type=="auc"){
   #we want highest AUC score, given the minimum thresholds
   bestWeights <- meetsMin[order(-meetsMin$AUC), ]
  }


  return(bestWeights[1, ])

  }
```

Models

Two non-parametric approaches will be used with this data: random forests and neural networks. Neither has underlying assumptions about the data. Both tend to be good at handling both categorical predictors and categorical response variables.

Random forests are uniquely suited to this data because many of the predictors show strong associations. A random forest model will choose different predictors in each of the trees in the forest, limiting the effects of the strong associations.

Neural networks tend to be powerful at finding subtle patterns in data. Our data does not have any particularly strong and obvious patterns, so a neural network may be a good approach.

```
#make a dataframe to store results from all our models
selResults <- data.frame(model=c('Random Forest Cutoff',  'Random Forest Strata',  'Neural Net', 'Elastic Net'),
             #general values - all models
             AUC=rep(NA, 4),
             Accuracy=rep(NA, 4),
             Threshold=rep(NA, 4),
             Sensitivity=rep(NA, 4),
             Specificity=rep(NA, 4),

             #random forest
             mtry=rep(NA, 4),
             #neural net
             caseWeight=rep(NA, 4),
             size=rep(NA, 4),
             decayRate=rep(NA, 4),
                        #penalized
                        lambda=rep(NA, 4),
                        alpha=rep(NA, 4)
                        )

#remove everything not needed
rm(cc,temp,trainIndex,c,cols,checkOtherCats)

## Warning in rm(cc, temp, trainIndex, c, cols, checkOtherCats): object 'cols'
## not found

## Warning in rm(cc, temp, trainIndex, c, cols, checkOtherCats): object
## 'checkOtherCats' not found
```

Random Forest Approaches for Imbalanced Data

Random forests contain two methods for handling imbalanced data. The first method is a cutoff threshold. The cutoff alters the voting strategy for each observation, switching from a "majority rule" approach to an approach that only requires our small class to get a percentage of votes representative of it's proportion of the data.

```
#number of columns
mtry = 18

#make a dataframe to store randomforest results for overall best
fResults <- data.frame('mtry'=seq(1:mtry), 'AUC' = rep(NA, mtry), 'Threshold' = rep(NA, mtry), 'Accuracy' = rep(NA, mtry)
, 'Sensitivity' = rep(NA, mtry), 'Specificity' = rep(NA, mtry))


#set the rare class prevalence
rare.class.prevalence = .06

#try the random forest at each mtry
for(m in 1:mtry){
  set.seed(13) #don't get consistent results unless I set the seed here


  #Balancing by voting rule, AUC of ROC will be unchanged...
  forest = randomForest(isUser~.,data=cc.train,
                 mtry=m,
                 importance=T,
                 cutoff=c(1-rare.class.prevalence,rare.class.prevalence)
                 )

  #get the probability that this is a user
  probYes <- forest$votes[, 2]

  #use the probabilities to tune for a best weighted threshold for accuracy
  bestWeights <- getBestWeightedThreshold(probYes, cc.train$isUser, type="accuracy")
  fResults$mtry[m] <- m
  fResults$AUC[m] <- bestWeights$AUC
  fResults$Threshold[m] <- bestWeights$threshold
  fResults$Accuracy[m] <- bestWeights$accuracy
  fResults$Specificity[m] <- bestWeights$specificity
  fResults$Sensitivity[m] <- bestWeights$sensitivity


  }


#get the best model by the accuracy metric
```

```
bestfResult <- fResults[order(-fResults$Accuracy), ]
rf.cutoff <- bestfResult[1, ]
selResults$AUC[1] <- rf.cutoff$AUC
selResults$Threshold[1] <- rf.cutoff$Threshold
selResults$Accuracy[1] <- rf.cutoff$Accuracy
selResults$Sensitivity[1] <- rf.cutoff$Sensitivity
selResults$Specificity[1] <- rf.cutoff$Specificity
selResults$mtry[1] <- rf.cutoff$mtry



#clean up
rm(bestfResult,bestWeights,fResults,rf.cutoff,forest,m,probYes,rare.class.prevalence,mtry)
```

An alternate approach in random forests is to do a stratified sampling within each tree. This approach down samples within each of the trees in the forest to have a more balanced dataset. Because we are building multiple trees, we will still use much of the majority class data, while allowing our model to learn to identify the minority class.

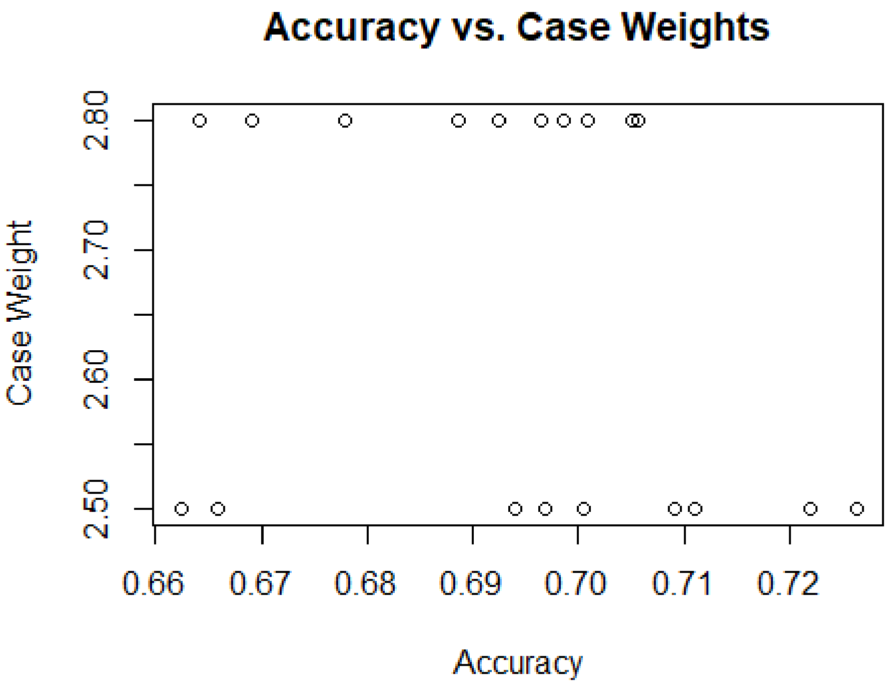
```
#number of columns
mtry = 18

#make a dataframe to store randomforest results for overall best
fResults <- data.frame('mtry'=seq(1:mtry), 'AUC' = rep(NA, mtry), 'Threshold' = rep(NA, mtry), 'Accuracy' = rep(NA, mtry)
, 'Sensitivity' = rep(NA, mtry), 'Specificity' = rep(NA, mtry))



rare.class.prevalence = .05

nRareSamples = 1000 * rare.class.prevalence

#try the random forest at each mtry
for(m in 1:mtry){
  set.seed(13) #don't get consistent results unless I set the seed here


  #Balancing by sampling stratification
  forest = randomForest(isUser~.,data=cc.train,
                  mtry=m,
                  importance=T,
                  strata=cc.train$isUser,
                  sampsize=c(nRareSamples,nRareSamples)
                  )

  #get the probability that this is a user
  probYes <- forest$votes[, 2]

  #use the probabilities to tune for a best weighted threshold for accuracy
  bestWeights <- getBestWeightedThreshold(probYes, cc.train$isUser, type="accuracy")
  fResults$mtry[m] <- m
  fResults$AUC[m] <- bestWeights$AUC
  fResults$Threshold[m] <- bestWeights$threshold
  fResults$Accuracy[m] <- bestWeights$accuracy
  fResults$Specificity[m] <- bestWeights$specificity
  fResults$Sensitivity[m] <- bestWeights$sensitivity




  }
```

```
#get the best model by the accuarcy metric
bestfResult <- fResults[order(-fResults$Accuracy), ]
rf.strata <- bestfResult[1, ]
selResults$AUC[2] <- rf.strata$AUC
selResults$Threshold[2] <- rf.strata$Threshold
selResults$Accuracy[2] <- rf.strata$Accuracy
selResults$Sensitivity[2] <- rf.strata$Sensitivity
selResults$Specificity[2] <- rf.strata$Specificity
selResults$mtry[2] <- rf.strata$mtry



#clean up
tryCatch(
  rm(bestfResult,bestWeights,fResults,rf.strata,forest,m,probYes,rare.class.prevalence,nRareSamples,mtry)
)
```

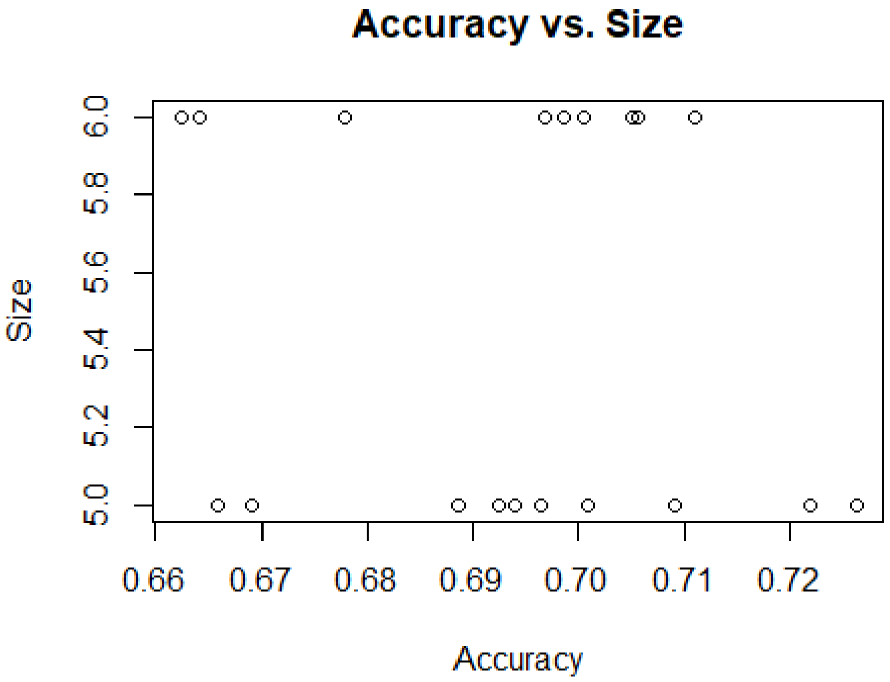
Artificial Neural Networks

Artificial Neural Networks do not rely on any underlying assumptions. They are perfect for data that does not have much separation, as they will attempt to find whatever pattern there is. However, with highly imbalanced data, we need to add another tuning parameter - a case weight to help balance out the dataset. Weights for all cases are 1 by default. We will leave our “No” weight at one and try a range of weights for our “Yes” values. Additionally, there are 2 other hyper-parameters to tune for neural nets - the size and the decay. All three parameters were tuned with a very large tuning grid, which was eventually scaled down to a small tuning grid that covered the best values for each parameter.

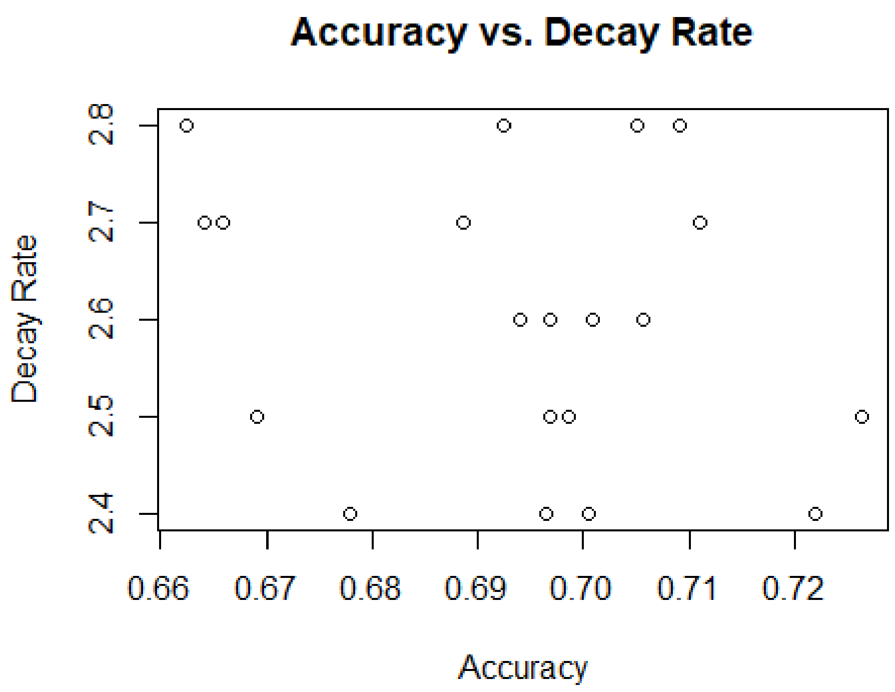
```
#set the seed
set.seed(13)
#get the number of rows
n = nrow(cc.train)
#set the number of folds
ncv = 10
#get the groups
if ((n%%ncv) == 0) {
        groups= rep(1:ncv,floor(n/ncv))} else {
        groups=c(rep(1:ncv,floor(n/ncv)),(1:(n%%ncv)))
        }

cvgroups = sample(groups,n)



#make a grid of values we're going to tune (these values were chosen based on initial CV runs with partial data)
tune.grid <- expand.grid(weight=c(2.5, 2.8),
                         size = c (5, 6),
                         decayRate = seq(2.4, 2.8, by = .1)
                         )

probs <- matrix( , nr = n, nc = nrow(tune.grid) )

for (i in 1:ncv){
    #define a group of true/false vectors for group i
    groupi = (cvgroups == i)

    for(t in 1:nrow(tune.grid)){

      #make a vector to weight the 'Yes' higher and 'No' lower
      case.weights <- rep(1, nrow(cc.train[!groupi, ]))
      case.weights[which(cc.train$isUser[!groupi] == 'Yes')] <- tune.grid$weight[t]

      #set seed again
      set.seed(13)
      #fit with this set of parameters
      nnet.fit <- nnet(isUser~.,
               data=cc.train[!groupi, ],
               size=tune.grid$size[t],
               decay=tune.grid$decayRate[t],
               maxit=1000,
               weights = case.weights,
               linout = F,
                   trace = F)

      thisPred <- predict(nnet.fit, newdata = cc.train[groupi, ], type="raw")
      probs[groupi, t] <- thisPred #seems to be predicting yes, which is alignment with the second level of the factor it's pred
icting

  } #end loop over tuning grid

}#end cv loop

cols <- nrow(tune.grid)
nResults <- data.frame('caseweight'=rep(NA,cols), 'size'=rep(NA,cols), 'decayRate'=rep(NA,cols),  'AUC' = rep(NA, cols)
, 'Threshold' = rep(NA, cols), 'Accuracy' = rep(NA, cols), 'Sensitivity' = rep(NA, cols), 'Specificity' = rep(NA, cols))




#get the best weights for each for each tuning grid result
for(w in 1:cols){
  #get best weights for accuracy
  bestWeights <- getBestWeightedThreshold(probs[, w], cc.train$isUser, type="accuracy")

  nResults$caseweight[w] <- tune.grid$weight[w]
  nResults$size[w] <- tune.grid$size[w]
  nResults$decayRate[w] <- tune.grid$decayRate[w]
  nResults$AUC[w] <- bestWeights$AUC[1]
  nResults$Threshold[w] <- bestWeights$threshold[1]
  nResults$Accuracy[w] <- bestWeights$accuracy[1]
  nResults$Specificity[w] <- bestWeights$specificity[1]
  nResults$Sensitivity[w] <- bestWeights$sensitivity[1]


}


plot(nResults$Accuracy, nResults$caseweight, ylab='Case Weight', xlab='Accuracy', main="Accuracy vs. Case Weights")
```



```
plot(nResults$Accuracy, nResults$size, ylab='Size', xlab='Accuracy', main="Accuracy vs. Size")
```



```
plot(nResults$Accuracy, nResults$decayRate, ylab='Decay Rate', xlab='Accuracy', main="Accuracy vs. Decay Rate")
```



```
#get the best overall and add it to the dataframe
ordered <- nResults[order(-nResults$Accuracy), ]
bestNet <- ordered[1, ]

selResults$AUC[3] <- bestNet$AUC
selResults$Threshold[3] <- bestNet$Threshold
selResults$Accuracy[3] <- bestNet$Accuracy
selResults$Sensitivity[3] <- bestNet$Sensitivity
selResults$Specificity[3] <- bestNet$Specificity
selResults$caseWeight[3] <- bestNet$caseweight
selResults$size[3] <- bestNet$size
selResults$decayRate[3] <- bestNet$decayRate
```

Reducing the variables

In consultation with the client, she requested that I consider models that reduced the overall number of predictors used. While the demographics questions are questions that are routinely included in application materials for educational programs, she felt that 11 risk questions might be too many, in conjunction with other questions being asked in a pretest environment. There are a few methods that might be worth considering when variable reduction is a concern. A pruned decision tree is an option. But, in initial testing, pruned decision trees performed very poorly (and consistently just predicted everyone as not an abuser). Another option is an elastic net or LASSO model. This data meets the general assumptions for a logistic model (the underlying model that elastic net uses) as outlined by Complete Dissertations (http://www.statisticssolutions.com/assumptions-of-logistic-regression/).

```
getBestPenalized <- function(data, lambdalist, cvgroups, drawPlot=FALSE) {

  fulldata = data  #only input the data used to fit the model
  #get inner x
  x = model.matrix(isUser~.,data=fulldata)[,-1]
  #get inner y
  y = fulldata$isUser
```

```
n = dim(fulldata)[1]

# calculate what fraction of the total each class has
fraction <- table(y)/length(y)
# assign 1 - that value to a "weights" vector
weights <- 1 - fraction[as.character(y)]

#PENALIZED cross-validation with a variety of alphas
alphaList = seq(.1, 1, by=.1)
#make a vector to track best alpha
alphaResult = data.frame('alpha' = alphaList,
                'AUC' = rep(NA, length(alphaList)),
                'Lambda' = rep(NA, length(alphaList)),
                'Threshold' =  rep(NA, length(alphaList)),
                'Accuracy' =  rep(NA, length(alphaList)),
                'Sensitivity' =  rep(NA, length(alphaList)),
                'Specificity' =  rep(NA, length(alphaList))
)

#loop through potential alphas and do CV with each one, tracking the best result
for(a in length(alphaList)){

  set.seed(13) #don't get consistent results unless I set the seed here
  glm = cv.glmnet(x, y,
           lambda=lambdalist,
           alpha = alphaList[a],
           nfolds=10,
           foldid=cvgroups,
           family="binomial",
           type.measure="auc",
           weights=weights,
           keep=T) #use the auc measure

  #make a vector to track best lambda result
  lResult = data.frame('alpha' = alphaList[a],
                'AUC' = rep(NA, length(lambdalist)),
                'Lambda' = rep(NA, length(lambdalist)),
                'Threshold' =  rep(NA, length(lambdalist)),
                'Accuracy' =  rep(NA, length(lambdalist)),
                'Sensitivity' =  rep(NA, length(lambdalist)),
                'Specificity' =  rep(NA, length(lambdalist))
  )

  for (j in 1:length(lambdalist)){
    #get the predictions for the best fit
    probYes <-glm$fit.preval[, j]
    #get the best weighted threshold
    bestWeights <- getBestWeightedThreshold(probYes, y, type="accuracy")
    lResult$alpha[j] <- alphaList[a]
    lResult$AUC[j] <- bestWeights$AUC
    lResult$Lambda[j] <- lambdalist[j]
    lResult$Threshold[j] <- bestWeights$threshold
    lResult$Accuracy[j] <- bestWeights$accuracy
    lResult$Specificity[j] <- bestWeights$specificity
    lResult$Sensitivity[j] <- bestWeights$sensitivity
  }



  #sort lResults by accuracy decending
  lResult <- lResult[order(-lResult$Accuracy), ]
  #put the winner of this alpha in the alpha result
```

```
      alphaResult$alpha[a] <- alphaList[a]
      alphaResult$AUC[a] <- lResult$AUC[1]
      alphaResult$Lambda[a] <- lResult$Lambda[1]
      alphaResult$Threshold[a] <- lResult$Threshold[1]
      alphaResult$Accuracy[a] <- lResult$Accuracy[1]
      alphaResult$Specificity[a] <- lResult$Specificity[1]
      alphaResult$Sensitivity[a] <- lResult$Sensitivity[1]



    #if requested, draw the plots
    if(drawPlot==TRUE){
      plot(glm$lambda,glm$cvm,type="l",lwd=2,col="red",xlab="lambda",ylab="CV(10) AUC", xlim=c(0,1), main=paste("Penalized Model for Alpha ", a, " Best AUC (", lResult$AUC[1], ")"))
      abline(v=lResult$Lambda[1])

    }
  }


  #sort the dataframe by highest overall score
  alphaResult <- alphaResult[order(-alphaResult$Accuracy), ]


  #figure out which was the best alpha and lambda, based on CV

  return(alphaResult[1, ])

}

set.seed(13)
#number of folds
ncv = 10
n = nrow(cc.train)
groups = c(rep(1:ncv,floor(n/ncv)),1:(n%%ncv))  #produces list of group labels
cvgroups = sample(groups,n)  #orders randomly
lambdalist = exp((1:-100)/10)

bestPen <- getBestPenalized(cc.train, lambdalist, cvgroups, drawPlot = F)



selResults$AUC[4] <- bestPen$AUC
selResults$Threshold[4] <- bestPen$Threshold
selResults$Accuracy[4] <- bestPen$Accuracy
selResults$Sensitivity[4] <- bestPen$Sensitivity
selResults$Specificity[4] <- bestPen$Specificity
selResults$lambda[4] <- bestPen$Lambda
selResults$alpha[4] <- bestPen$alpha
```

Model Select Results

The following shows the results of the model selection process, sorted from most accurate to least accurate.

```
ordered <- selResults[order(-selResults$Accuracy), ]
acc <- ordered[, c(1,2,3,5,6)]
pander(acc)
```

	model	AUC	Accuracy	Sensitivity	Specificity
2	Random Forest Strata	0.7281	0.7364	0.6005	0.7444

4	LASSO	0.7295	0.7352	0.6011	0.7431
3	Neural Net	0.7205	0.7263	0.6023	0.7336
1	Random Forest Cutoff	0.6806	0.6667	0.6059	0.6703

Model Winner

The model selection winner was the stratified random forest model. Now that the model has been selected, first we must fit it to the entire training set.

```
#Get all the winning model parameters
winner <- ordered[1, ]
pander(t(winner))
```

	2
model	Random Forest Strata
AUC	0.728052
Accuracy	0.7363923
Threshold	0.5462882
Sensitivity	0.6004799
Specificity	0.7444354
mtry	18
caseWeight	NA
size	NA
decayRate	NA
lambda	NA
alpha	NA

```
#rerun the winning stratified random forest model
rare.class.prevalence = .05
nRareSamples = 1000 * rare.class.prevalence
m=18
set.seed(13)
#Balancing by sampling stratification
forest = randomForest(isUser~.,data=cc.train,
                     mtry=m,
                     importance=T,
                     strata=cc.train$isUser,
                     sampsize=c(nRareSamples,nRareSamples)
                     )
#get the probability that this is a user
probYes <- forest$votes[, 2]

#Use the chosen threshold
threshold <- 0.5462882

#Review Training Prediction Accuracy
pred      <- factor( ifelse(probYes > threshold, "Yes", "No") )
caret::confusionMatrix(pred, cc.train$isUser, positive='Yes')

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  20970   666
##        Yes  7199  1001
##
##                Accuracy : 0.7364
##                  95% CI : (0.7314, 0.7414)
##     No Information Rate : 0.9441
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1213
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.60048
##             Specificity : 0.74444
##          Pos Pred Value : 0.12207
##          Neg Pred Value : 0.96922
##              Prevalence : 0.05587
##          Detection Rate : 0.03355
##    Detection Prevalence : 0.27484
##       Balanced Accuracy : 0.67246
##
##        'Positive' Class : Yes
##
```

Predicting with the Test Set

Test set prediction accuracy was slightly higher than during the model selection process, with an accuracy of 73.86%. Most of that gain came in sensitivity, with our model able to predict 61% of the individuals that were users as users. Note, however, that 3064 people were falsely predicted to be users, for a false positive rate of about 25%.

```
#predict the test set
testPred <- predict(forest, newdata=cc.test, type="prob")

#get the probability that this is a user
probYes <- testPred[, 2]

#Review Test Predication Accuracy
pred      <- factor( ifelse(probYes > threshold, "Yes", "No") )
caret::confusionMatrix(pred, cc.test$isUser, positive='Yes')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  9008  278
##        Yes 3064  436
##
##                Accuracy : 0.7386
##                  95% CI : (0.7309, 0.7462)
##     No Information Rate : 0.9442
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1258
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.61064
##             Specificity : 0.74619
##          Pos Pred Value : 0.12457
##          Neg Pred Value : 0.97006
##              Prevalence : 0.05584
##          Detection Rate : 0.03410
##    Detection Prevalence : 0.27374
##       Balanced Accuracy : 0.67842
##
```

```
##       'Positive' Class : Yes
##
```

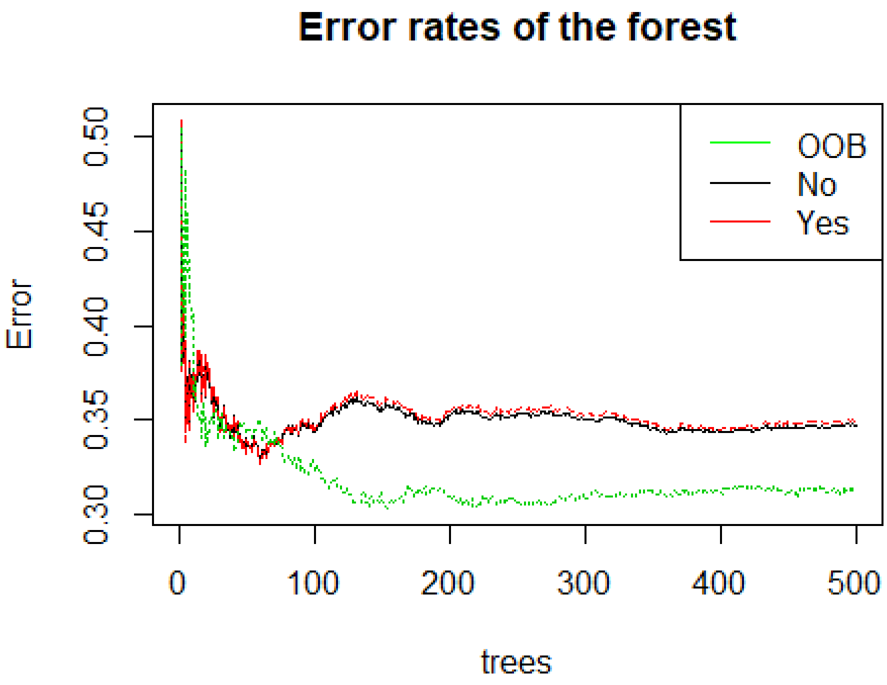
Understanding the Random Forest

Understanding a random forest can be challenging. The randomForestExplainer package can help visualize what is happening within the random forest. (https://cran.rstudio.com/web/packages/randomForestExplainer/vignettes/randomForestExplainer.html#introduction)

Forest Error

First, let’s review how the well the forest performed as the number of trees increased. Note that the out of box error rate appears to even out at about 200 trees and stays roughly the same, so our default of 500 trees appears to be plenty.

```r
plot(forest, main = "Error rates of the forest")
legend("topright", colnames(forest$err.rate), lty = c(1,1,1), col = c("green", "black", "red"))
```

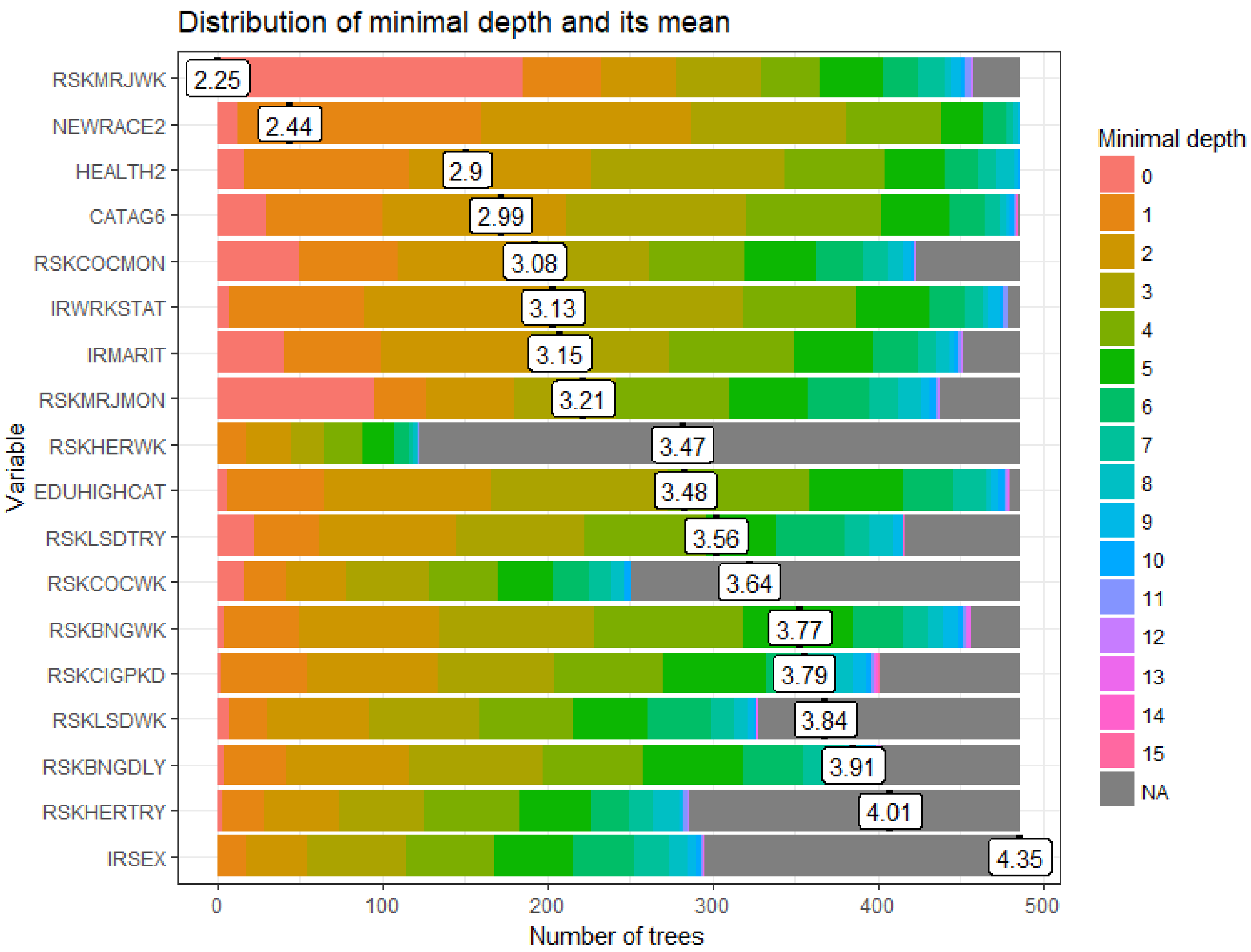


Minimal Depth Distribution

When thinking about trees in a random forest, one way to assess which variables were most important is to consider which variables were used “higher up” in the tree the most. A decision tree will always try to first split on the variable that provides the most discriminating information. There are multiple ways to review this. This view looks at only the relevant trees. In other words, it ignores the missing values if a variable is not included in a tree.

```r
#generated once and then loaded
#min_depth_frame <- min_depth_distribution(forest)
#save(min_depth_frame, file = "min_depth_frame.rda")
load("min_depth_frame.rda")
```

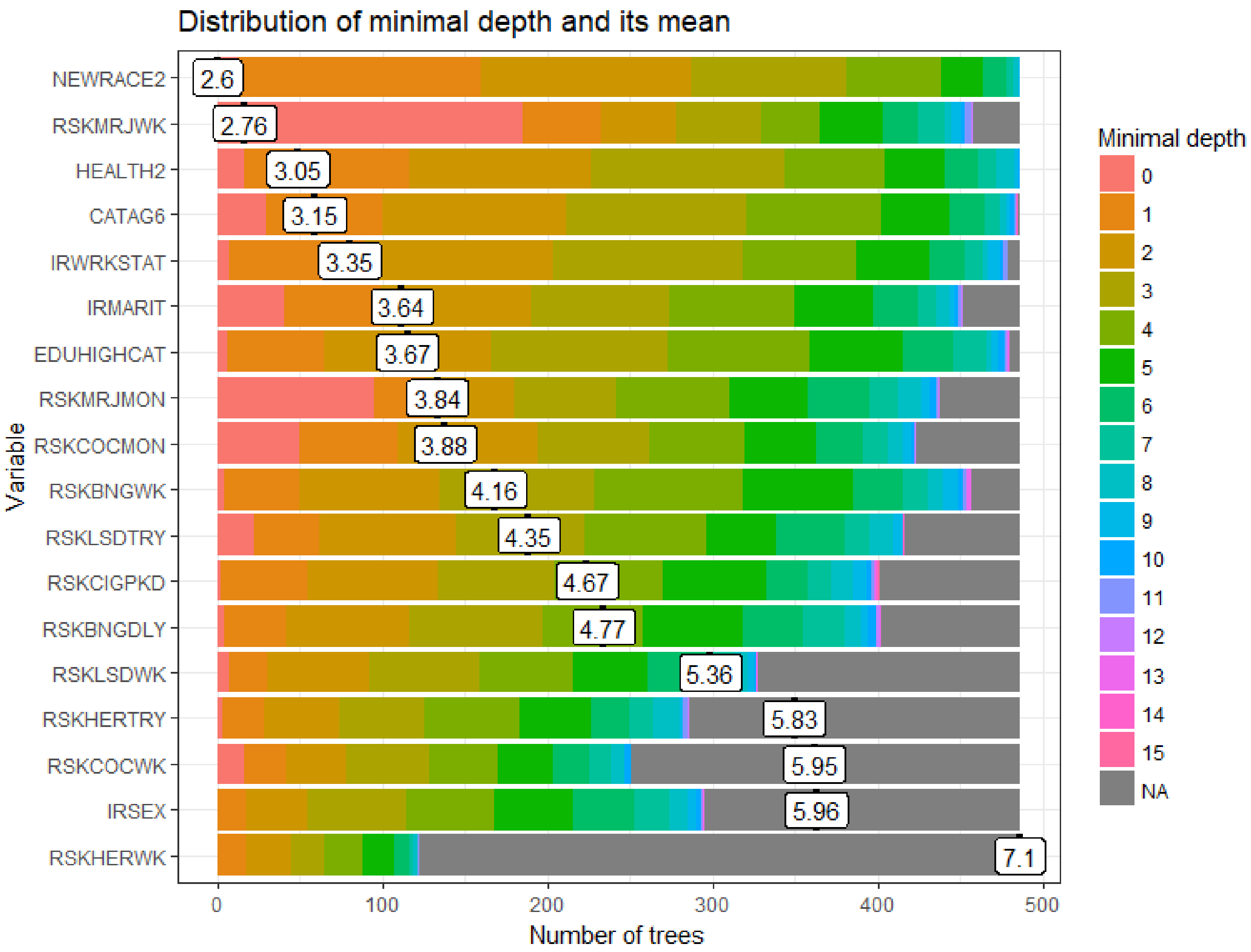
```
#plot the minimal depth and its mean
plot_min_depth_distribution(min_depth_frame, mean_sample = "relevant_trees", k = 18)
```



In this case, the risk of smoking marijuana weekly was the variable that had the smallest mean minimal depth (meaning it was used higher up in the tree more often). It was used as the first variable in approximately 200 trees. The risk of smoking marijuana monthly was the variable used as the first split in the next highest number of trees. But, here we see how random forests handle variables that provide similar information, because this variable has a much higher mean minimal depth. It provides similar information as our top variable, so is included in fewer trees, overall. Surprisingly, race appears to be an important variable as well, which was not clear from our initial data exploration.

We can also look at the mean minimal depth using all the trees, where missing values are filled in with the mean. This gives a slightly different perspective, with race have the smallest mean minimal depth.

```
#plot the minimal depth and its mean
plot_min_depth_distribution(min_depth_frame, mean_sample = "all_trees", k = 18)
```



Other Importance Measures

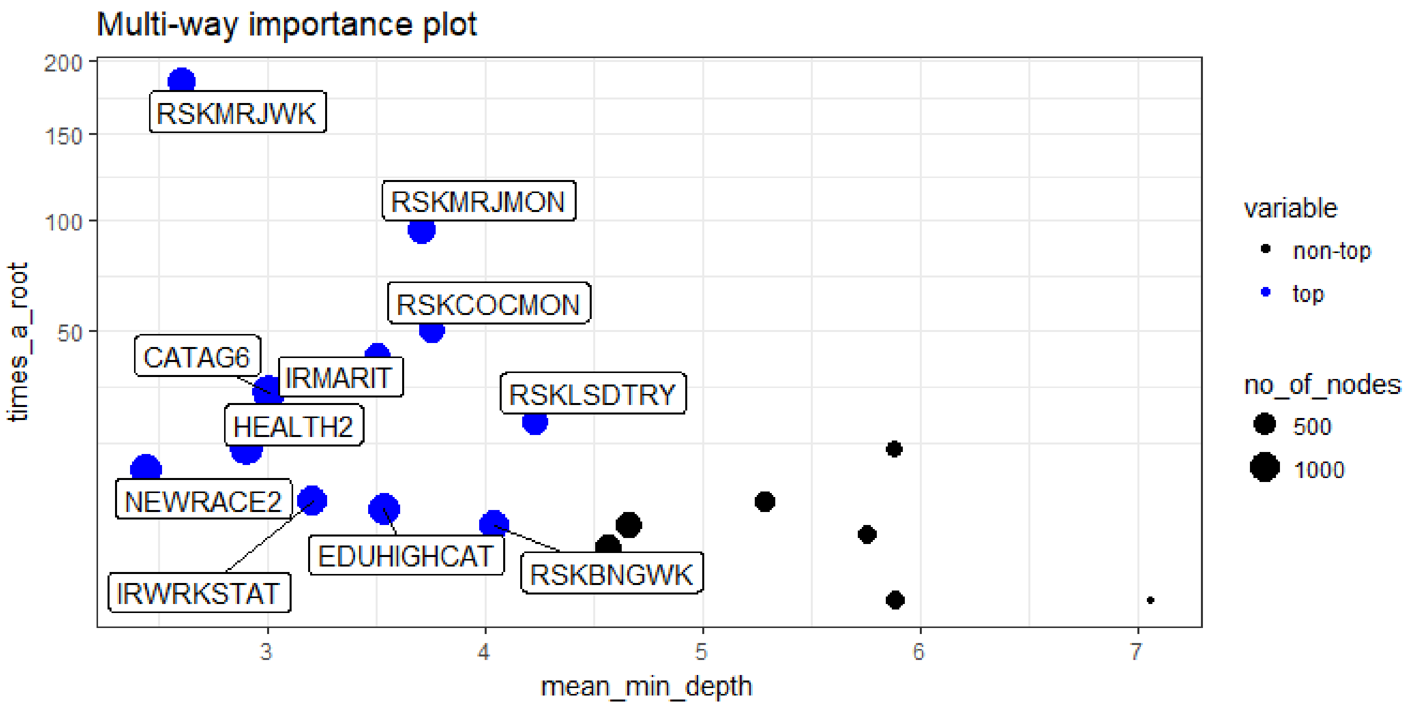
Mean minimal depth is not the only way to assess the importance of variables. We can also look at accuracy decrease, gini decrease, number of trees, number of nodes, and times at root, and a p-value statistic.

```
#get the variable importance frame
#importance_frame <- measure_importance(forest)
#save(importance_frame, file = "importance_frame.rda")
load("importance_frame.rda")
importance_frame[, -1] <- round(importance_frame[, -1], 3)
importance_frame

##      variable mean_min_depth no_of_nodes accuracy_decrease gini_decrease
## 1      CATAG6          3.003        1371             0.007         4.321
## 2  EDUHIGHCAT          3.534        1267             0.002         3.643
## 3     HEALTH2          2.903        1366             0.003         4.217
## 4     IRMARIT          3.504         777             0.002         2.933
## 5       IRSEX          5.890         400             0.001         1.231
## 6   IRWRKSTAT          3.203        1039             0.003         3.478
## 7    NEWRACE2          2.440        1176             0.002         4.356
## 8   RSKBNGDLY          4.666         735             0.002         2.166
## 9    RSKBNGWK          4.043        1034             0.002         2.929
## 10  RSKCIGPKD          4.570         744             0.003         2.288
## 11  RSKCOCMON          3.753         727             0.006         2.895
## 12   RSKCOCWK         5.880       324          0.002       1.175
## 13  RSKHERTRY         5.763       369          0.002       1.171
## 14   RSKHERWK         7.064       131          0.001       0.414
## 15  RSKLSDTRY         4.234       773          0.004       2.569
## 16   RSKLSDWK         5.281       489          0.003       1.533
## 17  RSKMRJMON         3.708       881          0.006       3.647
## 18   RSKMRJWK         2.598       960          0.007       5.016
##    no_of_trees times_a_root p_value
## 1          485           30   0.000
## 2          480            6   0.000
## 3          486           16   0.000
## 4          452           41   0.881
## 5          295            0   1.000
## 6          479            7   0.000
## 7          486           12   0.000
## 8          402            4   0.997
## 9          457            4   0.000
## 10         401            2   0.992
## 11         423           50   0.999
## 12         251           16   1.000
## 13         286            3   1.000
## 14         122            0   1.000
## 15         416           22   0.908
## 16         328            7   1.000
## 17         438           95   0.005
## 18         458          185   0.000
```

Times at root vs. mean minimal depth

Looking at a plot of the times at root versus the mean minimal depth, we can see that the risk of smoking marijuana weekly again appears to be a very important variable. (Variables in the top left of this plot are those of most importance.) Here we can see that race had a low mean minimal depth but was not used nearly as often at the root. That suggests that there are interactions going on between the variables used more often at the root (risk of smoking marijuana weekly and monthly, cocaine use monthly) that make it a more important variable once a prior split is made.

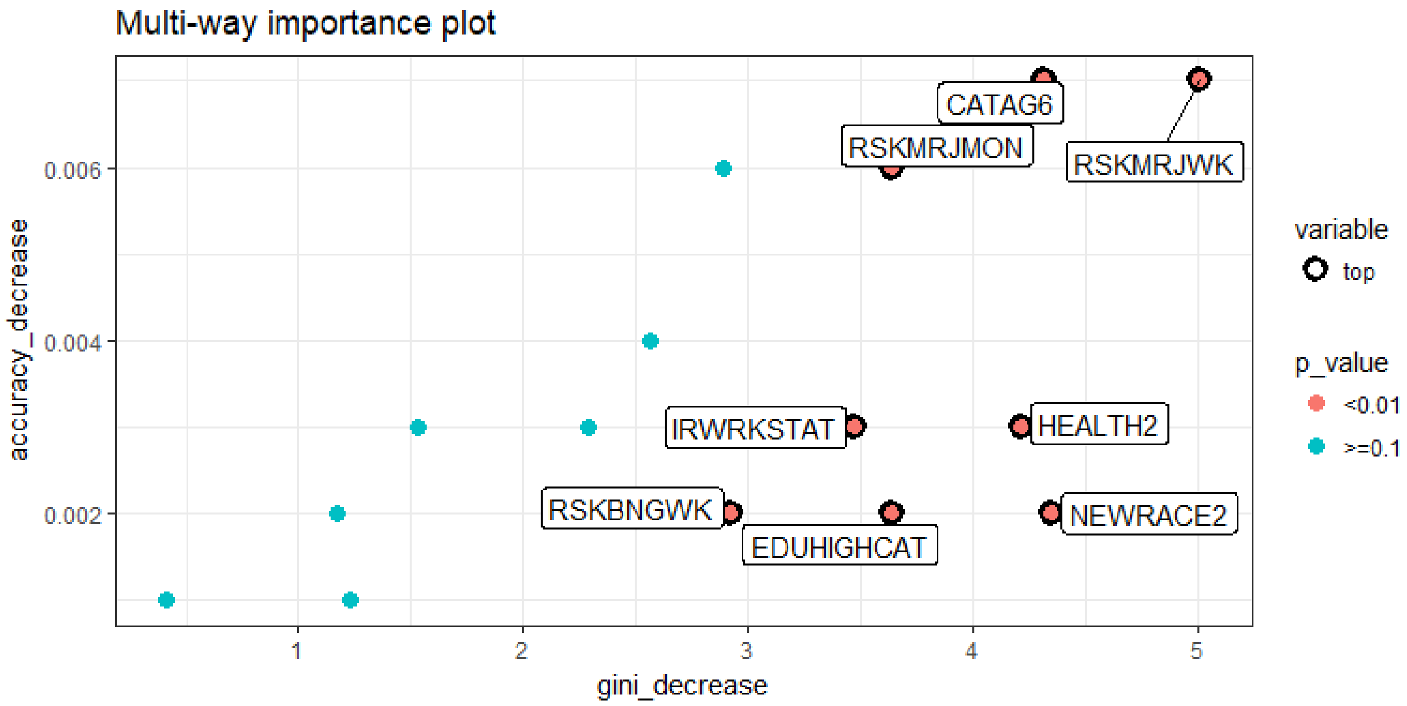
```
#get a plot of the top ten variables and their importance
plot_multi_way_importance(importance_frame, size_measure = "no_of_nodes")
```



Accuracy Decrease vs. Gini Decrease

In this plot, the most important variables are in the top right. Again, we see the risk of smoking marijuana being an important variable. In this view, which does not rely on where in the tree the variable was used, age appears to be a very important variable again, where race is a less important variable. It appears that there are 8 variables that are statistically significant, three of which are risk variables and the rest of which are demographic variables.

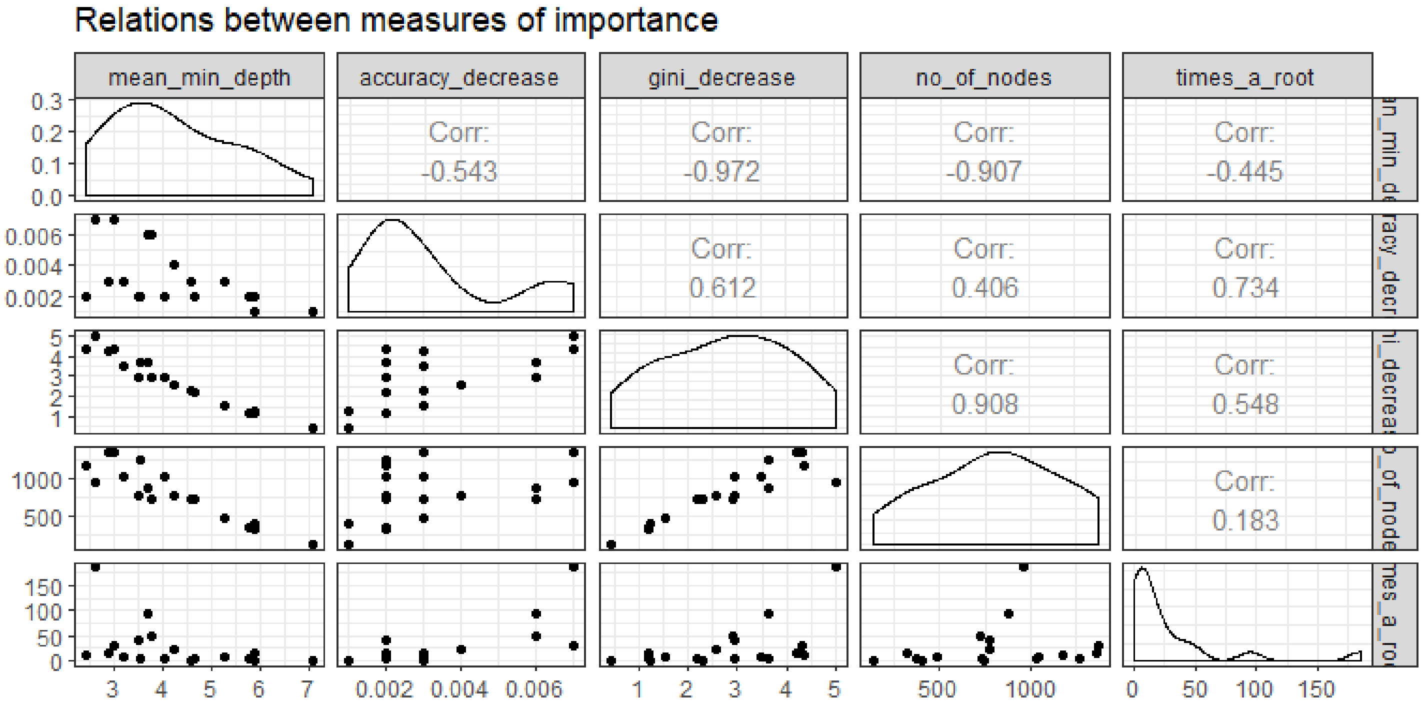
```
plot_multi_way_importance(importance_frame, x_measure = "gini_decrease", y_measure = "accuracy_decrease", size_measure = "p_value", no_of_labels = 8)
```



Correlations of importance measures

We can review a correlation matrix of importance measures to see if there are any 2 importance measures that are not highly correlated. We should review those as they may give us additional insight.

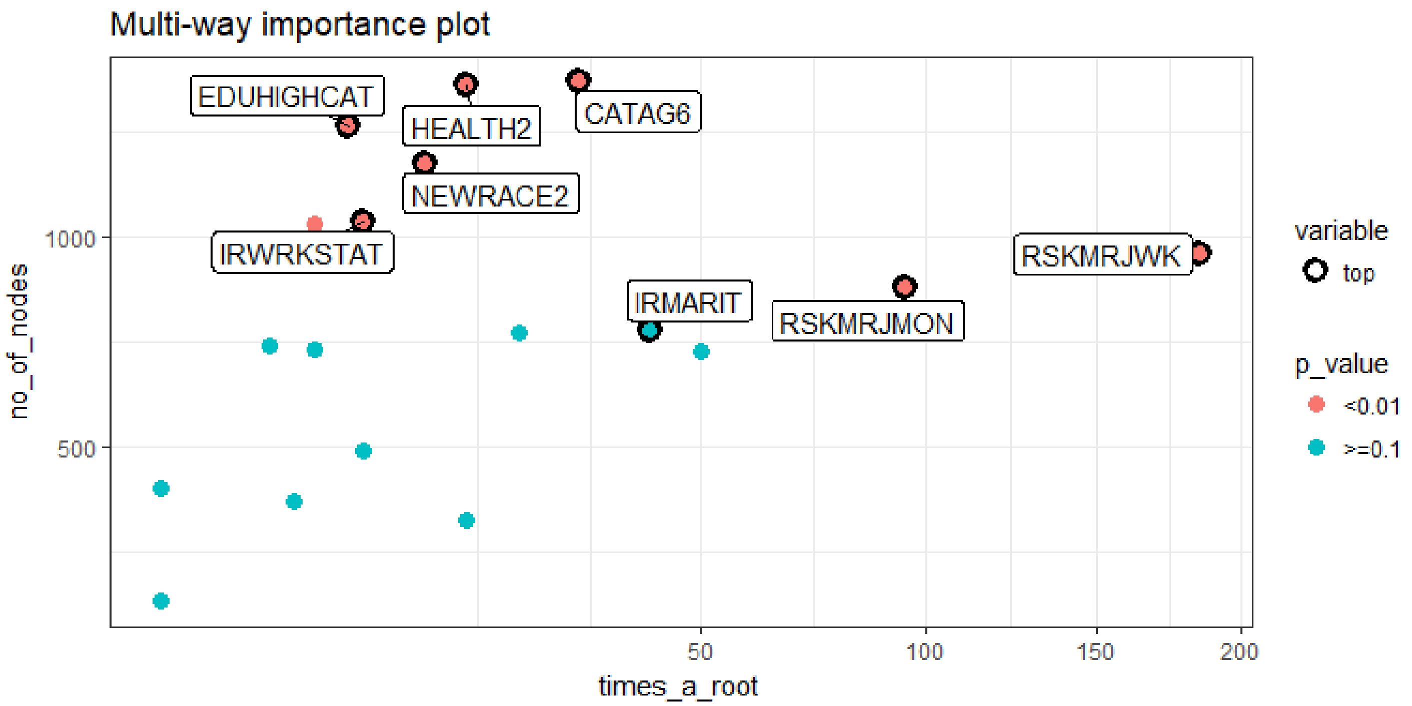
```
# Get relationships between importance measures (as each importance measure can show slightly different things)
plot_importance_ggpairs(importance_frame)
```



Measures with least correlation

The two measures that are least correlated with each other are the times at root and the number of nodes. In a plot of those two measures, the most important variables would be in the top right. The lack of correlation results in no variable in the top right corner of this graph. But, we can still see which are important in each of the two directions, and again we see the same variables as our top ones. But, note that marital status (which did not have a significant p-value) was considered a “top” variable here.

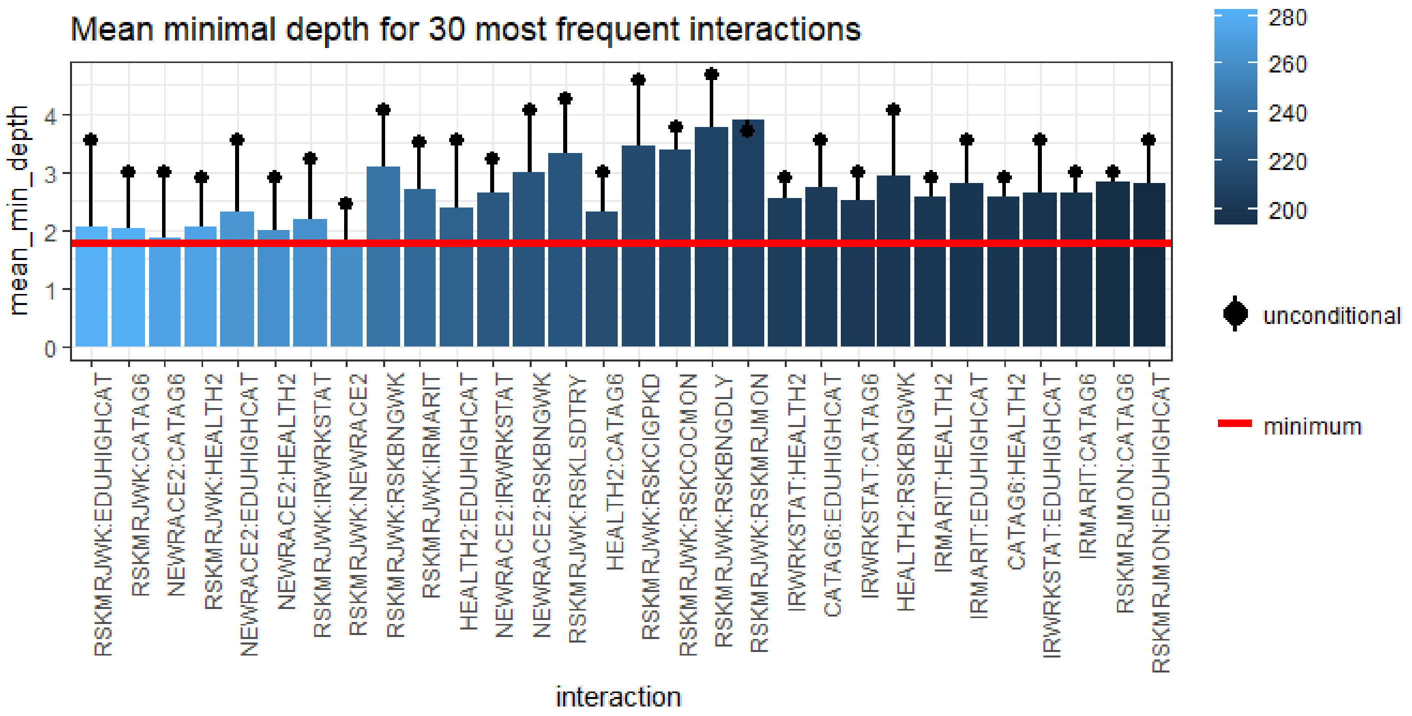
```
plot_multi_way_importance(importance_frame, x_measure = "times_a_root", y_measure = "no_of_nodes", size_measu
re = "p_value", no_of_labels = 8)
```



Reviewing interactions

Since we suspect that there may be interactions, we can review the interactions for out top 8 variables (those with a significant p-value). We'll assess this with the mean minimal depth and the number of trees.

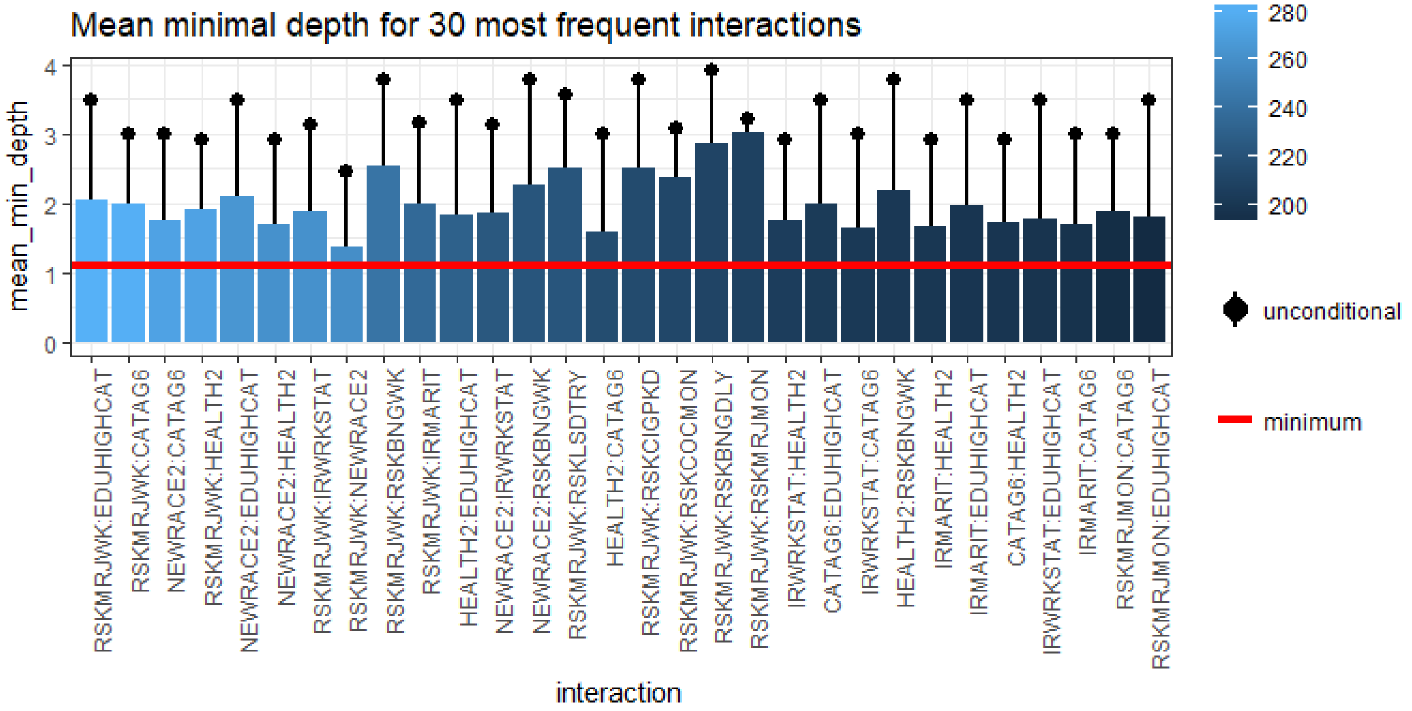
```
(vars <- important_variables(importance_frame, k = 8, measures = c("mean_min_depth", "no_of_trees")))

## [1] "CATAG6"     "EDUHIGHCAT" "HEALTH2"    "IRMARIT"    "IRWRKSTAT"
## [6] "NEWRACE2"   "RSKBNGWK"   "RSKMRJMON"  "RSKMRJWK"

 #interactions_frame <- min_depth_interactions(forest, vars)
 #save(interactions_frame, file = "interactions_frame.rda")
load("interactions_frame.rda")
head(interactions_frame[order(interactions_frame$occurrences, decreasing = TRUE), ])

##      variable root_variable mean_min_depth occurrences         interaction
## 18 EDUHIGHCAT      RSKMRJWK       2.053191         282 RSKMRJWK:EDUHIGHCAT
## 9      CATAG6      RSKMRJWK       2.015454         281     RSKMRJWK:CATAG6
## 6      CATAG6      NEWRACE2       1.876806         273     NEWRACE2:CATAG6
## 27    HEALTH2      RSKMRJWK       2.050435         273    RSKMRJWK:HEALTH2
## 15 EDUHIGHCAT      NEWRACE2       2.307883         265 NEWRACE2:EDUHIGHCAT
## 24    HEALTH2      NEWRACE2       1.989041         262    NEWRACE2:HEALTH2
##    uncond_mean_min_depth
## 18              3.534198
## 9               3.002613
## 6               3.002613
## 27              2.903292
## 15              3.534198
## 24              2.903292

#visualize the interactions
p <- plot_min_depth_interactions(interactions_frame)
p + theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



This plot is ordered by decreasing number of occurrences. So, the risk of smoking marijuana weekly is most often combined nearest the root of the tree with education level, then with age. Interestingly, when race is at the root, it is most often combined with age, and not any of the risk variables.

The above plot penalizes the variables that aren't used in a lot of trees. We can switch to a top trees view to remove that penalty.

```
# interactions_frame <- min_depth_interactions(forest, vars, mean_sample = "relevant_trees", uncond_mean_sample = "relevant_trees")
 #save(interactions_frame, file = "interactions_frame_relevant.rda")
load("interactions_frame_relevant.rda")
p <- plot_min_depth_interactions(interactions_frame)
p + theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



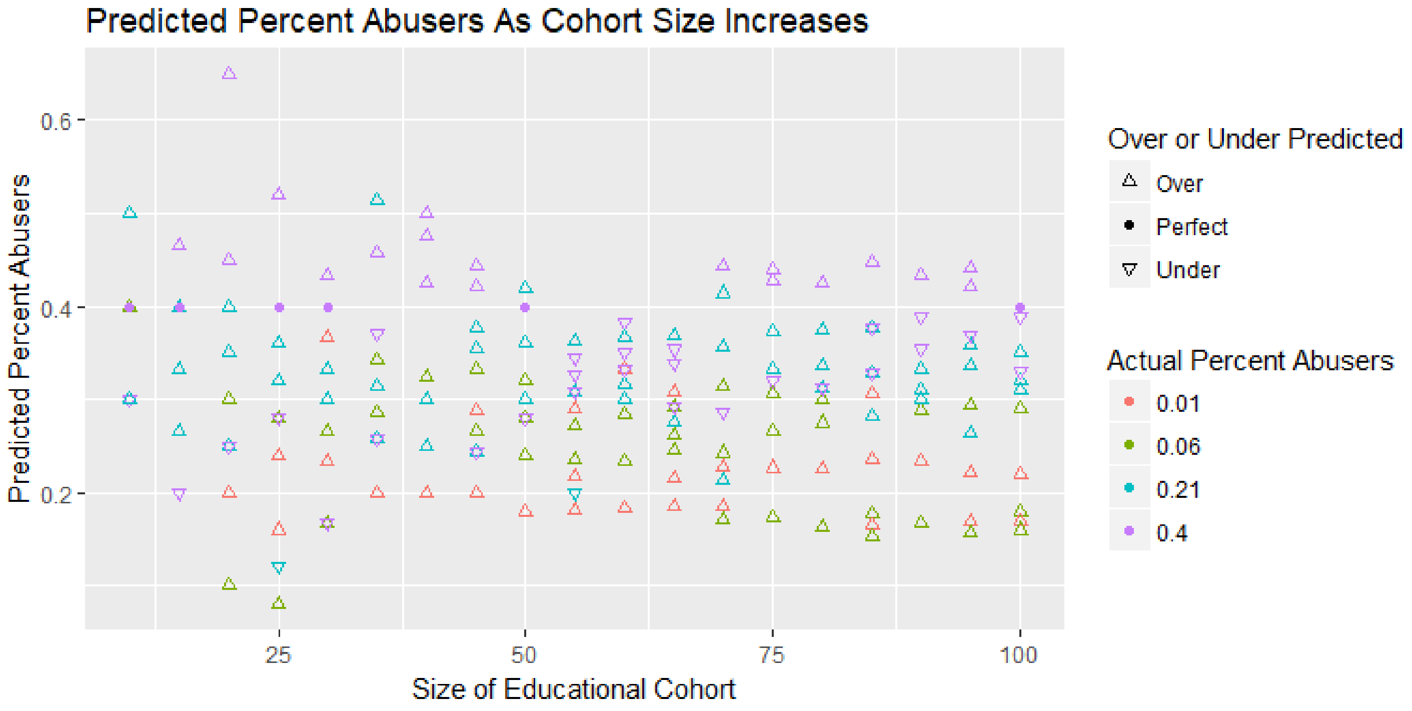
The results here are very similar and suggest that all of our top variables are used in most of the trees.

Putting our model into practice

In an educational setting, our student cohorts will be much smaller. Additionally, the estimated rate of opioid use disorder in Wisconsin is 21% (DHS, 2018). Let's review how our model does with smaller sets of our test observations, with various prevalence rates.

```
getPercentPredicted <- function(size,prev,threshold,forest,seed=1){
  #set a seed
  set.seed(seed)

  #get our separate datasets
  nonUsers <- cc.test[which(cc.test$isUser == 'No'), ]
  users <- cc.test[which(cc.test$isUser == 'Yes'), ]

  #get the number of abuser and non-abuser records we need
  userCount <- ceiling(size * prev)
  nonUserCount <- size-userCount

  #sample each and make a new dataframe
  userIndex <- sample(1:nrow(users), userCount)
  nonUserIndex <- sample(1:nrow(nonUsers), nonUserCount)

  #get our new data
  newTest <- rbind(nonUsers[nonUserIndex, ], users[userIndex, ])
  #randomly order it
  newTest <- newTest[sample(nrow(newTest)), ]

  #predict the test set
  newPred <- predict(forest, newdata=newTest, type="prob")

  #get the probability that this is a user
  probYes <- newPred[, 2]

  #Review Test Predication Accuracy
  pred      <- factor( ifelse(probYes > threshold, "Yes", "No") )

  tab <-  table(pred)
  #print(tab)

  percentPredicted <- tab[2]/size

  return(percentPredicted)

}

getPercentPredicted(65, .21, threshold,forest)

##       Yes
## 0.4307692

predGrid <- expand.grid(size=seq(10, 100, by=5),
                        perc=c(.01, .06, .21, .4)
                        )

n = nrow(predGrid)



#make a dataframe to store results
```

```
  predResults <- data.frame(size=rep(NA, n*3), predictedPercent=rep(NA, n*3), actualPercent=rep(NA, n*3))


#fill the dataframe
for (s in c(0,n,n*2)){
  for (r in 1:n){

  predictedPercent <- getPercentPredicted(predGrid$size[r], predGrid$perc[r], threshold,forest,s)
  predResults$size[r+s] <- predGrid$size[r]
  predResults$predictedPercent[r+s] <- predictedPercent
  predResults$actualPercent[r+s] <- predGrid$perc[r]
}

}

predResults$overunder <- rep('Under', nrow(predResults))
predResults$overunder[which(predResults$predictedPercent > predResults$actualPercent)] <- "Over"
predResults$overunder[which(predResults$predictedPercent == predResults$actualPercent)] <- "Perfect"

ggplot(predResults, aes(x=size, y=predictedPercent, color=as.factor(actualPercent), shape=as.factor(overunder))) +
  geom_point() +
  ggtitle("Predicted Percent Abusers As Cohort Size Increases") +
  guides(color=guide_legend(title="Actual Percent Abusers"), shape=guide_legend(title="Over or Under Predicted")) +
  ylab("Predicted Percent Abusers") +
  xlab("Size of Educational Cohort") +
  scale_shape_manual(values = c(24, 16, 25))
```



```
twentyone <- predResults[which(predResults$actualPercent == .21), ]
min(twentyone$predictedPercent)

## [1] 0.12

max(twentyone$predictedPercent)

## [1] 0.5142857

mean(twentyone$predictedPercent)

## [1] 0.3308987
```

```
sd(twentyone$predictedPercent)

## [1] 0.06780886
```

Testing Results

Our testing results show a couple of trends. First, if the actual percent of abusers is quite low (1% and 6%), there's a decreasing trend in predictions as cohort size increases. But in both cases, the predictions are generally consistently above the actual levels. When the level of actual abusers matches the state of Wisconsin abuser rate (21%), we see most of the predictions fairly tightly clustered, with a mean predicted percent of 33% and a standard deviation of 7%. In the case of a truly high percent of abusers (40%), we see much more variability in the predictions, with both over and under predictions. However, that is the only percentage for which we also saw perfect predictions.

In general, I think we can assume that the model overpredicts, especially with small cohort sizes. But, the actual amount of overprediction is variable. The best recommendation would be to use caution in implementing this model, as the accuracy and consistency at small sample sizes is just not there. If the model must be implemented, it would be best to use it as a "rule out" model instead of a "rule in" model. If the model predicts very low percentages of abusers, it is probably safe to assume the cohort does not contain many at risk individuals.