Speed Dating Partnership Analysis

## 1.Data Visulization

#(1)What are people looking for in their match?  
  
Speed\_Dating\_Data <- read\_csv("Speed Dating Data 1.csv")

## Parsed with column specification:  
## cols(  
## .default = col\_double(),  
## positin1 = col\_logical(),  
## field = col\_character(),  
## undergra = col\_logical(),  
## mn\_sat = col\_logical(),  
## tuition = col\_logical(),  
## from = col\_character(),  
## zipcode = col\_number(),  
## income = col\_number(),  
## career = col\_character(),  
## attr4\_1 = col\_logical(),  
## sinc4\_1 = col\_logical(),  
## intel4\_1 = col\_logical(),  
## fun4\_1 = col\_logical(),  
## amb4\_1 = col\_logical(),  
## shar4\_1 = col\_logical(),  
## attr5\_1 = col\_logical(),  
## sinc5\_1 = col\_logical(),  
## intel5\_1 = col\_logical(),  
## fun5\_1 = col\_logical(),  
## amb5\_1 = col\_logical()  
## # ... with 58 more columns  
## )

## See spec(...) for full column specifications.

## Warning: 271723 parsing failures.  
## row col expected actual file  
## 1795 positin1 1/0/T/F/TRUE/FALSE 2 'Speed Dating Data 1.csv'  
## 1795 attr4\_1 1/0/T/F/TRUE/FALSE 10 'Speed Dating Data 1.csv'  
## 1795 sinc4\_1 1/0/T/F/TRUE/FALSE 7 'Speed Dating Data 1.csv'  
## 1795 intel4\_1 1/0/T/F/TRUE/FALSE 7 'Speed Dating Data 1.csv'  
## 1795 fun4\_1 1/0/T/F/TRUE/FALSE 7 'Speed Dating Data 1.csv'  
## .... ........ .................. ...... .........................  
## See problems(...) for more details.

#First, data is checked for consistency since some of the participants will place the ranks differently than others (on a 1-10 scale compared to using a distribution of 100 points).   
  
#take related attributes with iid and gender into new data frame  
  
at11<-  
Speed\_Dating\_Data%>%  
 group\_by(gender) %>%  
 select(iid, gender, attr1\_1, sinc1\_1, intel1\_1, fun1\_1, amb1\_1, shar1\_1) %>%   
 unique()  
  
#Next, we would like to turn all NA into 0, but before this, we check if any entries in iid or gender is NA to prevent mislabels  
  
sum(is.na(at11$iid))

## [1] 0

sum(is.na(at11$gender))

## [1] 0

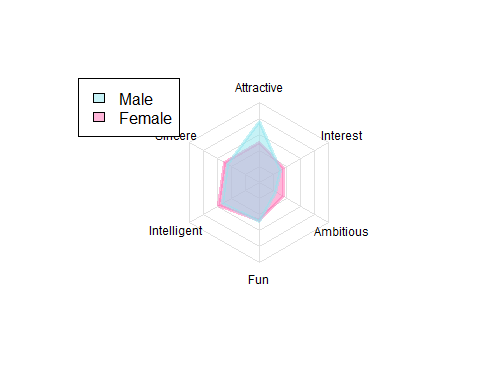
#Apply command to chnage all NA to 0  
  
at11[is.na(at11)] <- 0  
  
#Add column to check if total of attributions add up to 100  
  
at11$total <- rowSums(at11[,c("attr1\_1", "sinc1\_1", "intel1\_1", "fun1\_1", "amb1\_1", "shar1\_1")])  
  
table(at11$total)

##   
## 90 95 99.98 99.99 100 100.01 100.02 101 110 120   
## 4 1 2 19 470 24 9 1 1 2   
## 148   
## 1

#A total of 0 means all entries are missing and row is dropped  
  
at11<-  
at11 %>%   
 filter(!total == "0")  
  
#As there are entry errors in the data, all points are redistributed and curved to fit 100 total points  
  
at11$attr1\_1 <- round(at11$attr1\_1/at11$total\*100, digits = 2)  
at11$sinc1\_1 <- round(at11$sinc1\_1/at11$total\*100, digits = 2)  
at11$intel1\_1 <- round(at11$intel1\_1/at11$total\*100, digits = 2)  
at11$fun1\_1 <- round(at11$fun1\_1/at11$total\*100, digits = 2)  
at11$amb1\_1 <- round(at11$amb1\_1/at11$total\*100, digits = 2)  
at11$shar1\_1 <- round(at11$shar1\_1/at11$total\*100, digits = 2)  
  
at11$total <- rowSums(at11[,c("attr1\_1", "sinc1\_1", "intel1\_1", "fun1\_1", "amb1\_1", "shar1\_1")])  
  
at11$total <- round(at11$total, digits = 0)  
table(at11$total)

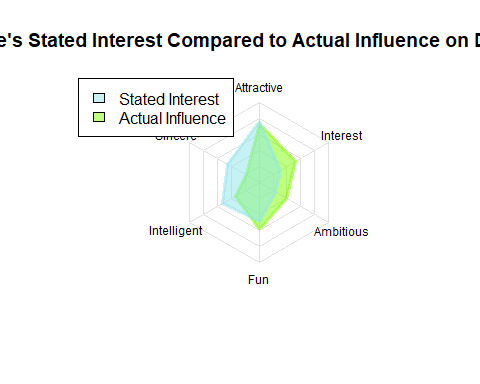
##   
## 100   
## 534

#Next, data is visualized using a radar chart  
  
test1 <-  
at11 %>%  
 group\_by(gender) %>%  
 summarise(Attractive = mean(attr1\_1), Sincere = mean(sinc1\_1), Intelligent = mean(intel1\_1), Fun = mean(fun1\_1), Ambitious = mean(amb1\_1), Interest = mean(shar1\_1))  
  
test1forplot <-  
test1 %>%   
 select(-gender)  
   
maxmin <- data.frame(  
 Attractive = c(36, 0),  
 Sincere = c(36, 0),  
 Intelligent = c(36, 0),  
 Fun = c(36, 0),  
 Ambitious = c(36, 0),  
 Interest = c(36, 0))  
  
test11 <- rbind(maxmin, test1forplot)  
  
test11male <- test11[c(1,2,4),]  
test11female <- test11[c(1,2,3),]  
  
radarchart(test11,  
 pty = 32,  
 axistype = 0,  
 pcol = c(adjustcolor("hotpink1", 0.5), adjustcolor("cadetblue2", 0.5)),  
 pfcol = c(adjustcolor("hotpink1", 0.5), adjustcolor("cadetblue2", 0.5)),  
 plty = 1,  
 plwd = 3,  
 cglty = 1,  
 cglcol = "gray88",  
 centerzero = TRUE,  
 seg = 5,  
 vlcex = 0.75,  
 palcex = 0.75)  
  
legend("topleft",   
 c("Male", "Female"),  
 fill = c(adjustcolor("cadetblue2", 0.5), adjustcolor("hotpink1", 0.5)))

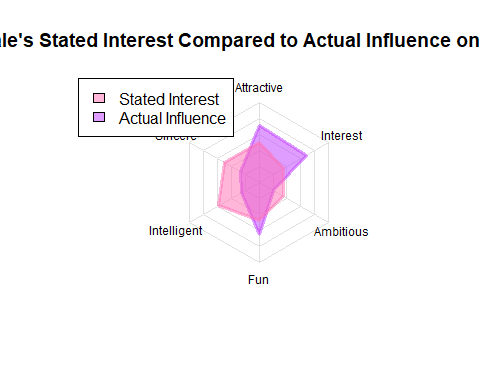


corforgraph2 <-data.frame(Traits = c("Average", "Male", "Female"),  
 corAttractive = c(coratr, cormatr, corfatr),  
 corSincere = c(corsin, cormsin, corfsin),  
 corIntelligence = c(corint, cormint, corfint),  
 corFun = c(corfun, cormfun, corffun),  
 corAmbitious = c(coramb, cormamb, corfamb),  
 corInterest = c(corshar, cormshar, corfshar))  
fin <- corforgraph2  
  
fin$total <- rowSums(fin[,c("corAttractive", "corSincere", "corIntelligence", "corFun", "corAmbitious", "corInterest")])  
  
fin$corAttractive <- round(fin$corAttractive/fin$total\*100, digits = 2)  
fin$corSincere <- round(fin$corSincere/fin$total\*100, digits = 2)  
fin$corIntelligence <- round(fin$corIntelligence/fin$total\*100, digits = 2)  
fin$corFun <- round(fin$corFun/fin$total\*100, digits = 2)  
fin$corAmbitious <- round(fin$corAmbitious/fin$total\*100, digits = 2)  
fin$corInterest <- round(fin$corInterest/fin$total\*100, digits = 2)  
  
fin <-  
fin %>%   
 select(corAttractive, corSincere, corIntelligence, corFun, corAmbitious, corInterest)  
  
colnames(fin) <- c("Attractive","Sincere", "Intelligent", "Fun", "Ambitious", "Interest")  
testn <- rbind(maxmin, fin, test1forplot)

testnmale <- testn[-c(3, 4, 6), ]  
testnfemale <- testn[-c(3, 5, 7), ]   
  
radarchart(testnmale,  
 pty = 32,  
 axistype = 0,  
 pcol = c(adjustcolor("chartreuse1", 0.5), adjustcolor("cadetblue2", 0.5)),  
 pfcol = c(adjustcolor("chartreuse1", 0.5), adjustcolor("cadetblue2", 0.5)),  
 plty = 1,  
 plwd = 3,  
 cglty = 1,  
 cglcol = "gray88",  
 centerzero = TRUE,  
 seg = 5,  
 vlcex = 0.75,  
 palcex = 0.75,  
 title = "Male's Stated Interest Compared to Actual Influence on Decision")  
  
legend("topleft",   
 c("Stated Interest", "Actual Influence"),  
 fill = c(adjustcolor("cadetblue2", 0.5), adjustcolor("chartreuse1", 0.5)))



radarchart(testnfemale,  
 pty = 32,  
 axistype = 0,  
 pcol = c(adjustcolor("darkorchid1", 0.5), adjustcolor("hotpink1", 0.5)),  
 pfcol = c(adjustcolor("darkorchid1", 0.5), adjustcolor("hotpink1", 0.5)),  
 plty = 1,  
 plwd = 3,  
 cglty = 1,  
 cglcol = "gray88",  
 centerzero = TRUE,  
 seg = 5,  
 vlcex = 0.75,  
 palcex = 0.75,  
 title = "Female's Stated Interest Compared to Actual Influence on Decision")  
  
legend("topleft",   
 c("Stated Interest", "Actual Influence"),  
 fill = c(adjustcolor("hotpink1", 0.5), adjustcolor("darkorchid1", 0.5)))



## 2.Data set collection

#setwd()  
rm(list = ls())  
Speed\_Dating\_Data <- read\_csv("Speed Dating Data 1.csv")

## Parsed with column specification:  
## cols(  
## .default = col\_double(),  
## positin1 = col\_logical(),  
## field = col\_character(),  
## undergra = col\_logical(),  
## mn\_sat = col\_logical(),  
## tuition = col\_logical(),  
## from = col\_character(),  
## zipcode = col\_number(),  
## income = col\_number(),  
## career = col\_character(),  
## attr4\_1 = col\_logical(),  
## sinc4\_1 = col\_logical(),  
## intel4\_1 = col\_logical(),  
## fun4\_1 = col\_logical(),  
## amb4\_1 = col\_logical(),  
## shar4\_1 = col\_logical(),  
## attr5\_1 = col\_logical(),  
## sinc5\_1 = col\_logical(),  
## intel5\_1 = col\_logical(),  
## fun5\_1 = col\_logical(),  
## amb5\_1 = col\_logical()  
## # ... with 58 more columns  
## )

## See spec(...) for full column specifications.

## Warning: 271723 parsing failures.  
## row col expected actual file  
## 1795 positin1 1/0/T/F/TRUE/FALSE 2 'Speed Dating Data 1.csv'  
## 1795 attr4\_1 1/0/T/F/TRUE/FALSE 10 'Speed Dating Data 1.csv'  
## 1795 sinc4\_1 1/0/T/F/TRUE/FALSE 7 'Speed Dating Data 1.csv'  
## 1795 intel4\_1 1/0/T/F/TRUE/FALSE 7 'Speed Dating Data 1.csv'  
## 1795 fun4\_1 1/0/T/F/TRUE/FALSE 7 'Speed Dating Data 1.csv'  
## .... ........ .................. ...... .........................  
## See problems(...) for more details.

data = Speed\_Dating\_Data[,c("iid","gender","pid","match","int\_corr","samerace",  
 "attr\_o","sinc\_o","intel\_o","fun\_o","amb\_o","shar\_o","like\_o",  
 "age","field\_cd","race","imprace","from","goal",  
 "date","go\_out","attr1\_1","sinc1\_1","intel1\_1","fun1\_1",  
 "amb1\_1","shar1\_1","match\_es")]  
class(data$match) <- "character"  
#check how many NAs in each column  
for (i in colnames(data)) {  
 print(paste(i,sum(is.na(data[,c(i)]))))  
}

## [1] "iid 0"  
## [1] "gender 0"  
## [1] "pid 10"  
## [1] "match 0"  
## [1] "int\_corr 75"  
## [1] "samerace 0"  
## [1] "attr\_o 199"  
## [1] "sinc\_o 268"  
## [1] "intel\_o 290"  
## [1] "fun\_o 341"  
## [1] "amb\_o 695"  
## [1] "shar\_o 1043"  
## [1] "like\_o 236"  
## [1] "age 32"  
## [1] "field\_cd 19"  
## [1] "race 0"  
## [1] "imprace 0"  
## [1] "from 21"  
## [1] "goal 0"  
## [1] "date 18"  
## [1] "go\_out 0"  
## [1] "attr1\_1 0"  
## [1] "sinc1\_1 0"  
## [1] "intel1\_1 0"  
## [1] "fun1\_1 10"  
## [1] "amb1\_1 20"  
## [1] "shar1\_1 42"  
## [1] "match\_es 1163"

### the ways to deal with missing value

# Just omit  
nrow(na.omit(data)) #after omitting na, 5735 rows left.

## [1] 5735

a = na.omit(data)

## 3. Data Cleaning

### 3.1 Partner-rated attributes score

#Calculate the mean of partner-rated attributes score  
b = aggregate(a[,c("attr\_o","sinc\_o","intel\_o",  
 "fun\_o","amb\_o","shar\_o","like\_o")], list(a$iid), mean)  
  
#combine dataframe  
a[,c("attr\_o","sinc\_o","intel\_o","fun\_o","amb\_o","shar\_o","like\_o")] = NULL  
names(b) = c("iid","attr\_m","sinc\_m","intel\_m","fun\_m","amb\_m","shar\_m","like\_m")  
a = merge(a,b,by="iid")

### 3.2 Region

#precleaned in excel file

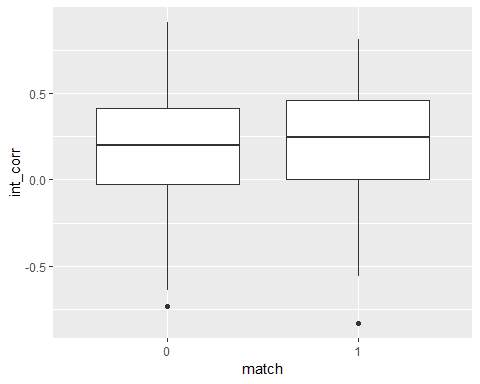
### 3.3 Most valued attributes

#Calculate the most valued attributes  
a$max1 = NA  
a$max2 = NA  
for (i in 1:nrow(a)) {  
 a[i,"max1"] =order(a[i,c("attr1\_1","sinc1\_1","intel1\_1",  
 "fun1\_1","amb1\_1","shar1\_1")], decreasing = T)[1]  
 a[i,"max2"] =order(a[i,c("attr1\_1","sinc1\_1","intel1\_1",  
 "fun1\_1","amb1\_1","shar1\_1")], decreasing = T)[2]  
}  
a[,c("attr1\_1","sinc1\_1","intel1\_1",  
 "fun1\_1","amb1\_1","shar1\_1")] = NULL  
#write.csv(a,file = "cleaned.csv")

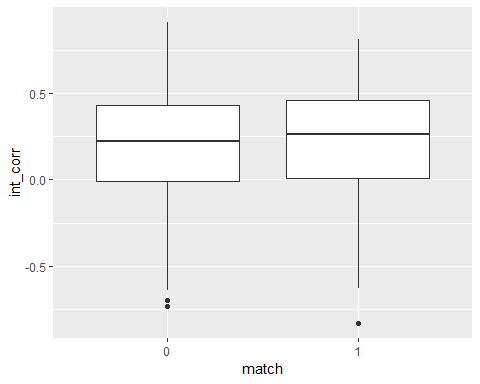
*The number in column max1,2 stands for 1-attr, 2-sinc, 3-intel, 4-fun, 5-amb, 6-shar.*

## 4. Descriptive Analysis

library(tidyr)  
library(ggplot2)  
  
#interest correlation vs match   
a%>%  
 filter(gender == "0")%>%  
 ggplot(aes(x = match,y = int\_corr))+geom\_boxplot()



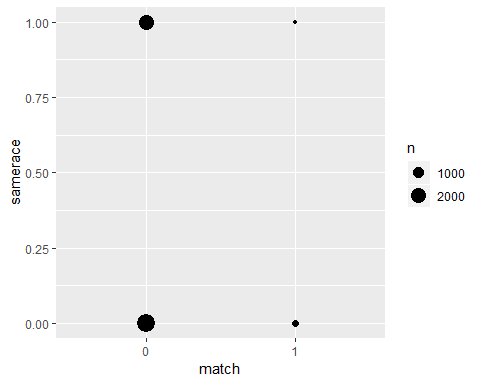
a%>%  
 filter(gender == "1")%>%  
 ggplot(aes(x = match,y = int\_corr))+geom\_boxplot()



fit <- aov(match ~ int\_corr, data=a)  
summary(fit)

## Df Sum Sq Mean Sq F value Pr(>F)   
## int\_corr 1 1.0 0.9537 6.971 0.00831 \*\*  
## Residuals 5733 784.3 0.1368   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

#(there is significant difference of the interest corrlation between match or not; higher correlation tends to have higher match probability.)  
  
#samerace vs match & gender  
a%>%  
 ggplot(aes(x = match,y = samerace))+geom\_count()



fit2 <- aov(match ~ samerace, data=a)  
summary(fit2)

## Df Sum Sq Mean Sq F value Pr(>F)  
## samerace 1 0.2 0.2220 1.621 0.203  
## Residuals 5733 785.0 0.1369

#(same race or not won't effect the match success rate)  
  
#attribute vs match & gender  
#done by excel, represent as radar chart  
  
#max attribute vs gender  
tb1 <- a%>% filter(gender=='0')  
 table(tb1$max1)

##   
## 1 2 3 4 5 6   
## 916 1077 724 161 20 67

table(tb1$max2)

##   
## 1 2 3 4 5 6   
## 374 735 1062 545 106 143

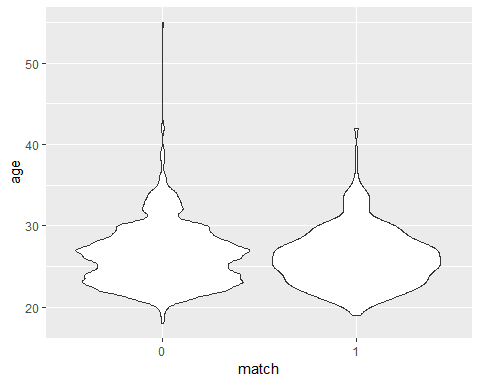
tb2 <- a%>% filter(gender=='1')  
 table(tb2$max1)

##   
## 1 2 3 4 5 6   
## 1857 320 391 122 15 65

table(tb2$max2)

##   
## 1 2 3 4 5 6   
## 450 937 888 340 9 146

#age vs match & gender  
ggplot(a,aes(match,age))+  
 geom\_violin()



fit3 <- aov(match~age,data = a)  
class(a$match) <- "character"  
class(a$match)

## [1] "character"

summary(fit3)

## Df Sum Sq Mean Sq F value Pr(>F)   
## age 1 0.4 0.4469 3.265 0.0708 .  
## Residuals 5733 784.8 0.1369   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

#age doesn't effect match rate  
  
#race & occupation (field) vs match & gender  
#done by tableu & excel  
  
class(a$match) <- "numeric"

## 5. Analysis of Demography

### 5.1 Cleaned the data for demography analysis

# different iid represent different people, only keep 1 record for each iid  
# Remove duplicates based on iid columns  
Dem\_data = data[!duplicated(data$iid), ]  
Dem\_data$match = as.factor(Dem\_data$match)

### 5.2 Use all the cleaned demography data and see general results

colnames(Dem\_data)

## [1] "iid" "gender" "pid" "match" "int\_corr" "samerace"  
## [7] "attr\_o" "sinc\_o" "intel\_o" "fun\_o" "amb\_o" "shar\_o"   
## [13] "like\_o" "age" "field\_cd" "race" "imprace" "from"   
## [19] "goal" "date" "go\_out" "attr1\_1" "sinc1\_1" "intel1\_1"  
## [25] "fun1\_1" "amb1\_1" "shar1\_1" "match\_es"

# Demography indicators include: race, age, field of career, imprace, goal, date and go out.  
glm\_general <- glm(match~samerace+age+field\_cd+race+goal+date+go\_out, data = Dem\_data, family = "binomial")  
  
summary(glm\_general)

##   
## Call:  
## glm(formula = match ~ samerace + age + field\_cd + race + goal +   
## date + go\_out, family = "binomial", data = Dem\_data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.0818 -0.6126 -0.5000 -0.3688 2.4119   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 1.43207 1.07496 1.332 0.18279   
## samerace -0.16170 0.26059 -0.621 0.53492   
## age -0.05507 0.03734 -1.475 0.14021   
## field\_cd -0.02571 0.03365 -0.764 0.44478   
## race 0.01208 0.09862 0.122 0.90253   
## goal 0.09394 0.08236 1.141 0.25404   
## date -0.23730 0.08798 -2.697 0.00699 \*\*  
## go\_out -0.30457 0.14610 -2.085 0.03710 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 446.33 on 529 degrees of freedom  
## Residual deviance: 423.07 on 522 degrees of freedom  
## (4 observations deleted due to missingness)  
## AIC: 439.07  
##   
## Number of Fisher Scoring iterations: 5

# Use "match" as target, only date and go\_out are statistically significant.

### 5.3 Change the numeric data into factor and do specific logit regression

# Change the numeric data into factor  
c <- transform(Dem\_data,field\_cd = as.factor(field\_cd), race = as.factor(race), goal = as.factor(goal), date = as.factor(date), go\_out = as.factor(go\_out))  
  
glm\_date = glm(match~ date,data=c,family = "binomial")  
summary(glm\_date)

##   
## Call:  
## glm(formula = match ~ date, family = "binomial", data = c)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.0579 -0.5721 -0.5063 -0.3912 2.2838   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.2877 0.7638 -0.377 0.70642   
## date2 -0.1978 0.8861 -0.223 0.82335   
## date3 -1.4395 0.8547 -1.684 0.09214 .   
## date4 -0.9601 0.7923 -1.212 0.22556   
## date5 -2.2437 0.8588 -2.613 0.00899 \*\*  
## date6 -1.7019 0.8089 -2.104 0.03539 \*   
## date7 -1.9577 0.8404 -2.330 0.01983 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 450.78 on 532 degrees of freedom  
## Residual deviance: 427.54 on 526 degrees of freedom  
## (1 observation deleted due to missingness)  
## AIC: 441.54  
##   
## Number of Fisher Scoring iterations: 5

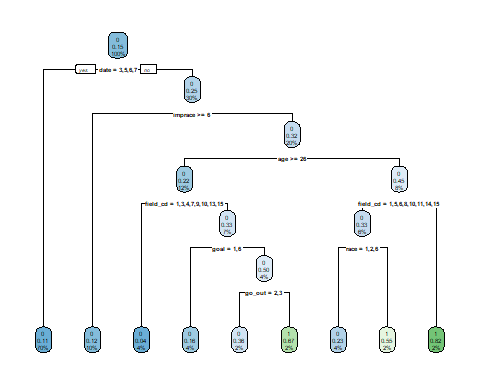
# date 5, 6, 7 are significant  
  
glm\_go\_out <- glm(match~go\_out, data=c, family = "binomial")  
summary(glm\_go\_out)

##   
## Call:  
## glm(formula = match ~ go\_out, family = "binomial", data = c)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -0.6870 -0.6870 -0.5373 -0.5373 2.5951   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.3236 0.1850 -7.155 8.38e-13 \*\*\*  
## go\_out2 -0.5390 0.2836 -1.900 0.0574 .   
## go\_out3 -0.5161 0.3200 -1.613 0.1068   
## go\_out4 -2.0086 1.0344 -1.942 0.0522 .   
## go\_out5 -15.2425 692.6889 -0.022 0.9824   
## go\_out6 -15.2425 1073.1090 -0.014 0.9887   
## go\_out7 -15.2425 1696.7344 -0.009 0.9928   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 451.11 on 533 degrees of freedom  
## Residual deviance: 435.66 on 527 degrees of freedom  
## AIC: 449.66  
##   
## Number of Fisher Scoring iterations: 15

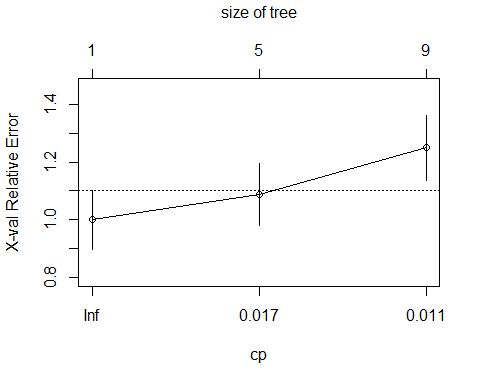
# go out = 1 is significant.

### 5.4 Classification tree

library(rpart)  
library(rpart.plot)  
ct\_Dem\_model <- rpart(match~race+age+imprace+field\_cd+goal+date+go\_out,data= c,  
 method="class", control = rpart.control(maxdepth = 7))  
rpart.plot(ct\_Dem\_model)



plotcp(ct\_Dem\_model)



printcp(ct\_Dem\_model)

##   
## Classification tree:  
## rpart(formula = match ~ race + age + imprace + field\_cd + goal +   
## date + go\_out, data = c, method = "class", control = rpart.control(maxdepth = 7))  
##   
## Variables actually used in tree construction:  
## [1] age date field\_cd go\_out goal imprace race   
##   
## Root node error: 80/534 = 0.14981  
##   
## n= 534   
##   
## CP nsplit rel error xerror xstd  
## 1 0.021875 0 1.0000 1.0000 0.10309  
## 2 0.012500 4 0.9125 1.0875 0.10667  
## 3 0.010000 8 0.8625 1.2500 0.11269

## 6. Analysis for random matching

### 6.1 dataset

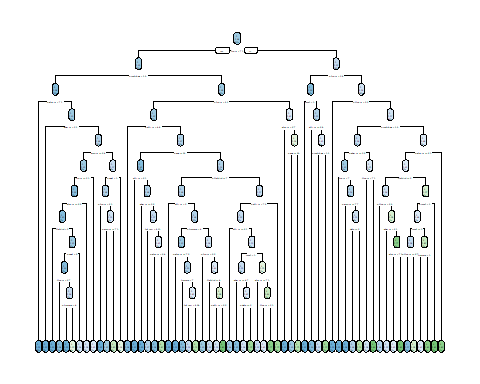
n = a[1,7:24]  
newname = paste("p\_",names(n), sep = "")  
p = setNames(data.frame(matrix(ncol = 18, nrow = 0)), newname)  
data2 = setNames(data.frame(matrix(ncol = 42,nrow = 0)),c(names(a),newname))  
for (i in 1:nrow(a)) {  
 partner\_id = a[i,"pid"]  
 p[i,] = a[a$iid==partner\_id,][1,7:24]  
}  
data2 = na.omit(cbind(a,p))  
data2 = data2[data2$gender=="0",]  
data2$max\_match = ifelse(data2$max1==data2$p\_max1,1,0)

### 5.2 classification tree

set.seed(1) # set a random seed   
index <- sample(nrow(data2), nrow(data2)\*0.2) # random selection of indices.   
test <- data2[index,] # save 20% as a test dataset  
training <-data2[-index,]

library(rpart)  
library(rpart.plot)  
ct\_model<-  
 rpart(match~max\_match+gender+race+imprace+int\_corr+samerace+field\_cd+goal+date+attr\_m+sinc\_m+intel\_m+fun\_m+amb\_m+shar\_m+like\_m+max1+max2+p\_age+p\_field\_cd+p\_race+p\_imprace+p\_goal+p\_date+p\_match\_es+p\_attr\_m+p\_sinc\_m+p\_intel\_m+p\_fun\_m+p\_amb\_m+p\_shar\_m+p\_like\_m+p\_max1+p\_max2,  
 data=training,  
 method="class",  
 control = rpart.control(cp=0))  
rpart.plot(ct\_model)

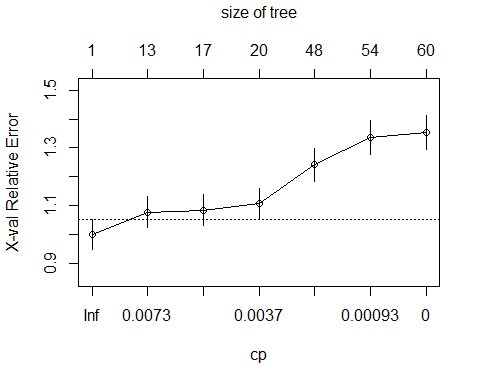
## Warning: labs do not fit even at cex 0.15, there may be some overplotting



printcp(ct\_model)

##   
## Classification tree:  
## rpart(formula = match ~ max\_match + gender + race + imprace +   
## int\_corr + samerace + field\_cd + goal + date + attr\_m + sinc\_m +   
## intel\_m + fun\_m + amb\_m + shar\_m + like\_m + max1 + max2 +   
## p\_age + p\_field\_cd + p\_race + p\_imprace + p\_goal + p\_date +   
## p\_match\_es + p\_attr\_m + p\_sinc\_m + p\_intel\_m + p\_fun\_m +   
## p\_amb\_m + p\_shar\_m + p\_like\_m + p\_max1 + p\_max2, data = training,   
## method = "class", control = rpart.control(cp = 0))  
##   
## Variables actually used in tree construction:  
## [1] amb\_m attr\_m date field\_cd fun\_m imprace   
## [7] int\_corr like\_m max1 max2 p\_age p\_amb\_m   
## [13] p\_attr\_m p\_date p\_field\_cd p\_fun\_m p\_goal p\_imprace   
## [19] p\_like\_m p\_match\_es p\_max1 p\_shar\_m p\_sinc\_m shar\_m   
## [25] sinc\_m   
##   
## Root node error: 310/1973 = 0.15712  
##   
## n= 1973   
##   
## CP nsplit rel error xerror xstd  
## 1 0.00829493 0 1.00000 1.0000 0.052144  
## 2 0.00645161 12 0.84194 1.0774 0.053733  
## 3 0.00430108 16 0.81613 1.0839 0.053860  
## 4 0.00322581 19 0.80323 1.1065 0.054302  
## 5 0.00161290 47 0.68387 1.2419 0.056785  
## 6 0.00053763 53 0.67419 1.3355 0.058344  
## 7 0.00000000 59 0.67097 1.3548 0.058652

plotcp(ct\_model)



#find the best tree with lowest xerror  
#min\_xerror\_tree<-ct\_model$cptable[which.min(ct\_model$cptable[,"xerror"]),]  
#min\_xerror\_tree  
#min\_xerror\_tree<-prune(ct\_model, cp=min\_xerror\_tree[1])  
#rpart.plot(min\_xerror\_tree)  
#the problem here is that the best is when nsplit=0, making the tree invalid  
  
#just use cp that is close to the smallest xerror  
ct\_model<-  
 rpart(match~max\_match+gender+race+imprace+int\_corr+samerace+field\_cd+goal+date+attr\_m+sinc\_m+intel\_m+fun\_m+amb\_m+shar\_m+like\_m+max1+max2+p\_age+p\_field\_cd+p\_race+p\_imprace+p\_goal+p\_date+p\_match\_es+p\_attr\_m+p\_sinc\_m+p\_intel\_m+p\_fun\_m+p\_amb\_m+p\_shar\_m+p\_like\_m+p\_max1+p\_max2,  
 data=training,  
 method="class",  
 control = rpart.control(cp=0.00829493))

### 5.3 CT quality exam

#choose the threshold  
for (i in 10:90) {  
 i = i/100  
 test$ct\_pred\_prob<-predict(ct\_model,test)[,2]  
 test$ct\_pred\_class=ifelse(test$ct\_pred\_prob>i,"Yes","No")  
 table <- table(test$ct\_pred\_class,test$match, dnn=c("predicted","actual"))   
# print(table)  
 print(paste(i,(table[1]+table[4])/493))  
}

## [1] "0.1 0.198782961460446"  
## [1] "0.11 0.198782961460446"  
## [1] "0.12 0.261663286004057"  
## [1] "0.13 0.811359026369168"  
## [1] "0.14 0.811359026369168"  
## [1] "0.15 0.811359026369168"  
## [1] "0.16 0.811359026369168"  
## [1] "0.17 0.811359026369168"  
## [1] "0.18 0.811359026369168"  
## [1] "0.19 0.815415821501014"  
## [1] "0.2 0.815415821501014"  
## [1] "0.21 0.815415821501014"  
## [1] "0.22 0.815415821501014"  
## [1] "0.23 0.815415821501014"  
## [1] "0.24 0.815415821501014"  
## [1] "0.25 0.815415821501014"  
## [1] "0.26 0.815415821501014"  
## [1] "0.27 0.815415821501014"  
## [1] "0.28 0.815415821501014"  
## [1] "0.29 0.817444219066937"  
## [1] "0.3 0.817444219066937"  
## [1] "0.31 0.817444219066937"  
## [1] "0.32 0.817444219066937"  
## [1] "0.33 0.817444219066937"  
## [1] "0.34 0.817444219066937"  
## [1] "0.35 0.817444219066937"  
## [1] "0.36 0.81947261663286"  
## [1] "0.37 0.81947261663286"  
## [1] "0.38 0.81947261663286"  
## [1] "0.39 0.81947261663286"  
## [1] "0.4 0.81947261663286"  
## [1] "0.41 0.81947261663286"  
## [1] "0.42 0.81947261663286"  
## [1] "0.43 0.81947261663286"  
## [1] "0.44 0.81947261663286"  
## [1] "0.45 0.81947261663286"  
## [1] "0.46 0.81947261663286"  
## [1] "0.47 0.81947261663286"  
## [1] "0.48 0.81947261663286"  
## [1] "0.49 0.81947261663286"  
## [1] "0.5 0.81947261663286"  
## [1] "0.51 0.81947261663286"  
## [1] "0.52 0.81947261663286"  
## [1] "0.53 0.81947261663286"  
## [1] "0.54 0.81947261663286"  
## [1] "0.55 0.81947261663286"  
## [1] "0.56 0.81947261663286"  
## [1] "0.57 0.81947261663286"  
## [1] "0.58 0.81947261663286"  
## [1] "0.59 0.81947261663286"  
## [1] "0.6 0.81947261663286"  
## [1] "0.61 0.81947261663286"  
## [1] "0.62 0.81947261663286"  
## [1] "0.63 0.81947261663286"  
## [1] "0.64 0.81947261663286"  
## [1] "0.65 0.81947261663286"  
## [1] "0.66 0.813387423935091"  
## [1] "0.67 0.813387423935091"  
## [1] "0.68 0.813387423935091"  
## [1] "0.69 0.813387423935091"  
## [1] "0.7 0.813387423935091"  
## [1] "0.71 0.813387423935091"  
## [1] "0.72 0.813387423935091"  
## [1] "0.73 0.813387423935091"  
## [1] "0.74 0.813387423935091"  
## [1] "0.75 0.813387423935091"  
## [1] "0.76 0.813387423935091"  
## [1] "0.77 0.813387423935091"  
## [1] "0.78 0.8052738336714"  
## [1] "0.79 0.8052738336714"  
## [1] "0.8 0.8052738336714"  
## [1] "0.81 0.8052738336714"  
## [1] "0.82 0.8052738336714"  
## [1] "0.83 0.8052738336714"  
## [1] "0.84 0.8052738336714"  
## [1] "0.85 0.807302231237322"  
## [1] "0.86 0.807302231237322"  
## [1] "0.87 0.807302231237322"  
## [1] "0.88 0.807302231237322"  
## [1] "0.89 0.807302231237322"  
## [1] "0.9 0.807302231237322"

#so when threshold = 0.5 (0.36-0.65), the prediction has highest correct rate, at 81.94%.

library(pROC)

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

## The following object is masked from 'package:fmsb':  
##   
## roc

## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

ct\_roc<-roc(test$match,test$ct\_pred\_prob,auc=TRUE)  
plot(ct\_roc,print.auc=TRUE,col="blue")

