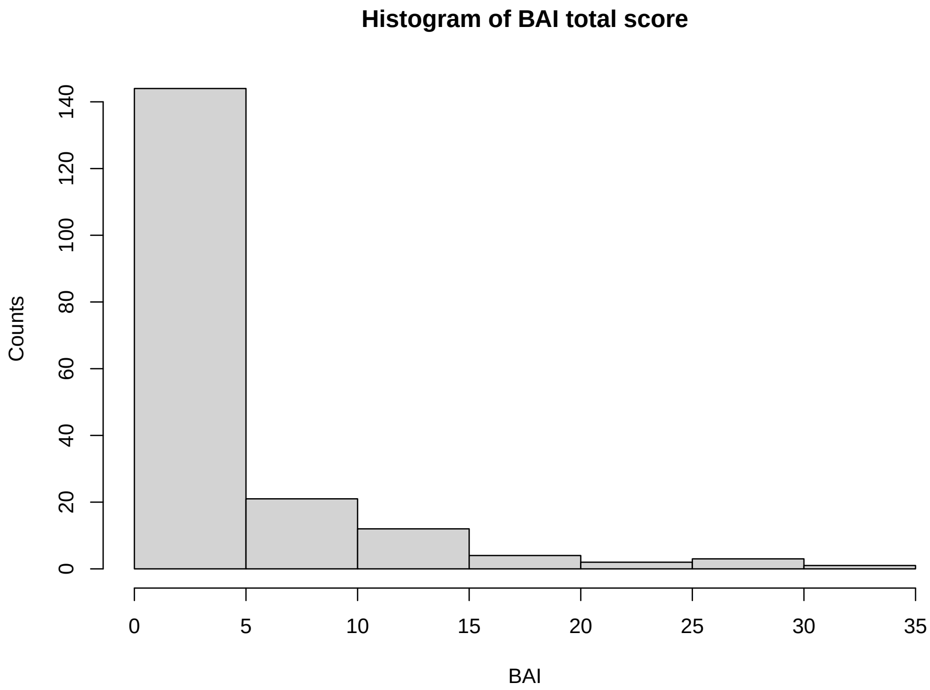
STAT 427 Milestone 2 Second Trial

03/27/2021, Shuyu Jia

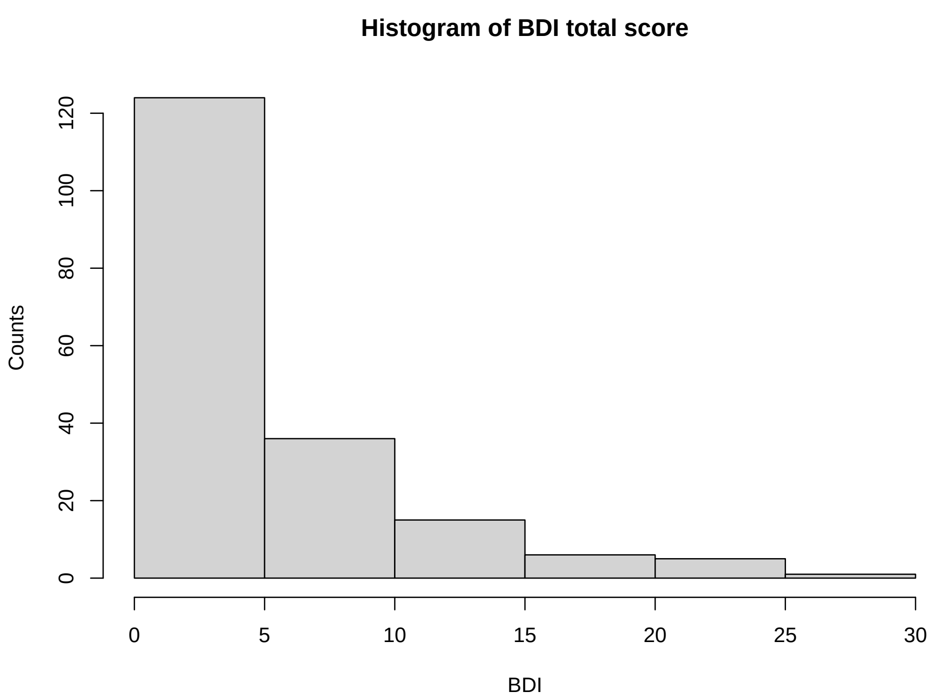
```r
# load packages
library(statmod)
library(tidyverse)
library(factoextra)
library(pheatmap)
library(umap)
library(glmnet)
library(caret)
library(ROSE)
library(randomForest)
library(rpart)
library(rpart.plot)
library(gbm)

# read in data
data = read.csv("updated_combined_dataset.csv",skip = 1)
```

```r
# extract emotional-related columns
emotion_data = data.matrix(data[,c(8,12,23,24,25)])
emotion_data = emotion_data[complete.cases(emotion_data),]
colnames(emotion_data) = c("THI_E", "TPFQ_E", "TFI_E", "BAI", "BDI")
emotion_df = data.frame(emotion_data)
```

```r
hist(emotion_data[,4], xlab = "BAI", ylab = "Counts", main = "Histogram of BAI total score")
```




```r
hist(emotion_data[,5], xlab = "BDI", ylab = "Counts", main = "Histogram of BDI total score")
```

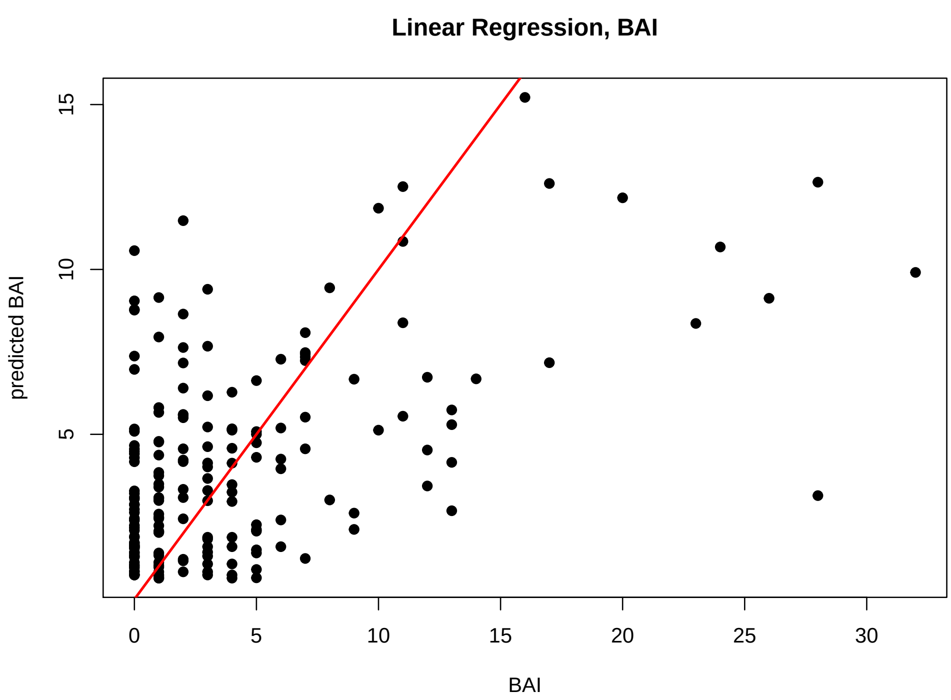


```r
# train-test split
set.seed(1)
tst_idx = sample(1:187, 37, replace = FALSE)
trn_emotion = emotion_df[-tst_idx,]
tst_emotion = emotion_df[tst_idx,]
```

```r
# linear regression, BAI response
lm_mod = lm(BAI ~ THI_E + TPFQ_E + TFI_E, data = emotion_df)
preds = predict(lm_mod, emotion_df[,1:3])

rmse = function(act, pred){
  sqrt(mean((act-pred)^2))
}

mae = function(act, pred){
  mean(abs(act-pred))
}

mae(emotion_df[,4], preds)
```

```
## [1] 3.291817
```

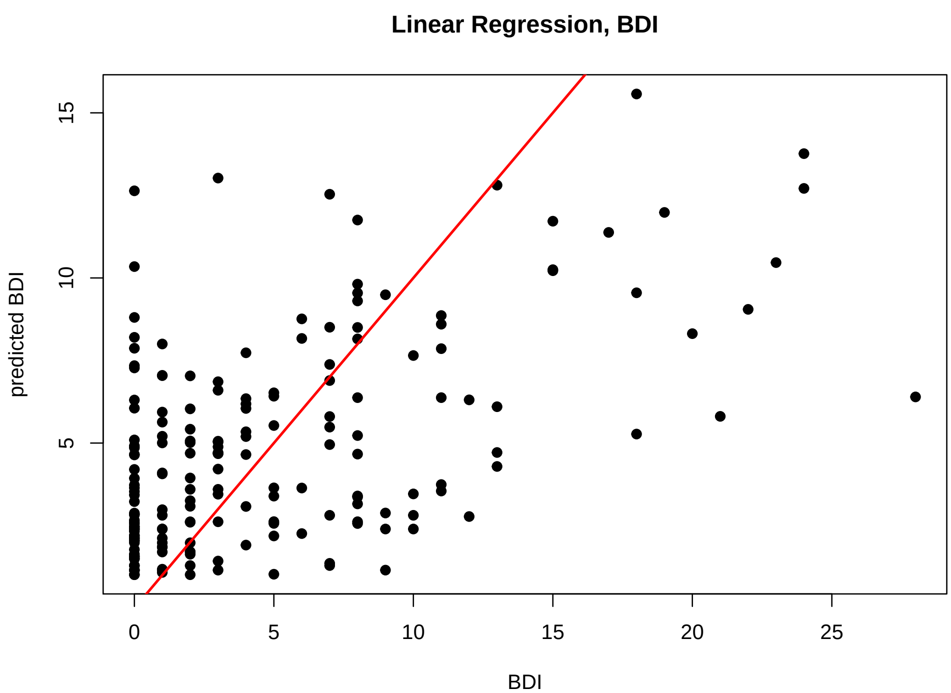
```r
plot(emotion_df[,4], preds, pch=19, main = "Linear Regression, BAI",
     xlab = "BAI", ylab = "predicted BAI")
abline(0,1, col="red", lwd = 2)
```



```r
# linear regression, BDI response
lm_mod_2 = lm(BDI ~ THI_E + TPFQ_E + TFI_E, data = emotion_df)
preds = predict(lm_mod_2, emotion_df[,1:3])
mae(emotion_df[,5], preds)
```

```
## [1] 3.614928
```

```r
plot(emotion_df[,5], preds, pch=19, main = "Linear Regression, BDI",
     xlab = "BDI", ylab = "predicted BDI")
abline(0,1, col="red", lwd = 2)
```

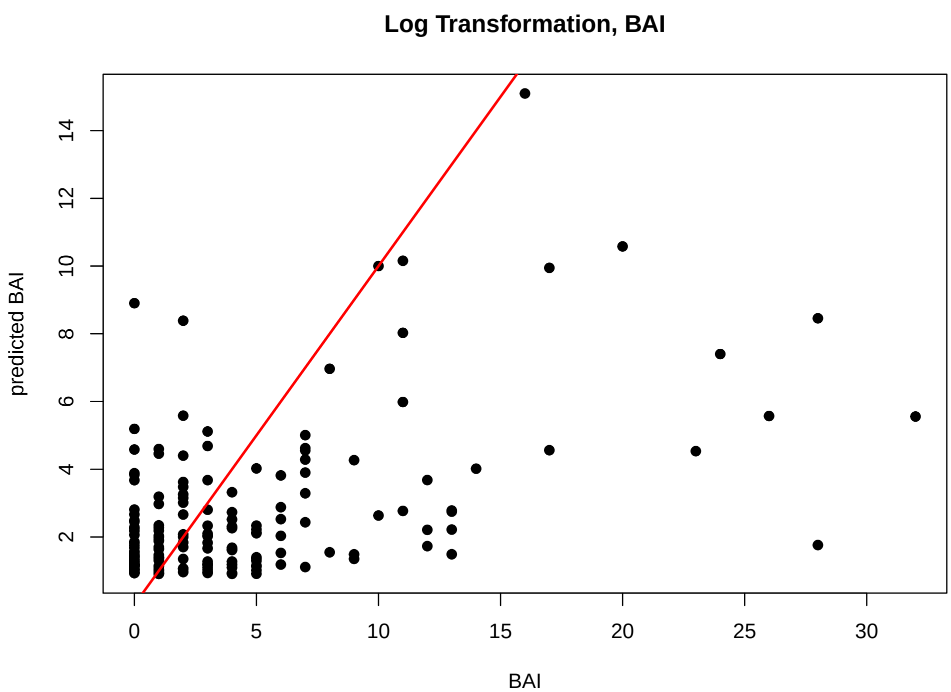


Linear regression is terrible.

```r
# log transform, BAI response
log_mod = lm(log(BAI+1) ~ THI_E + TPFQ_E + TFI_E, data = emotion_df)
preds = predict(log_mod, emotion_df[,1:3])
mae(emotion_df[,4], exp(preds)-1)
```

```
## [1] 3.095514
```

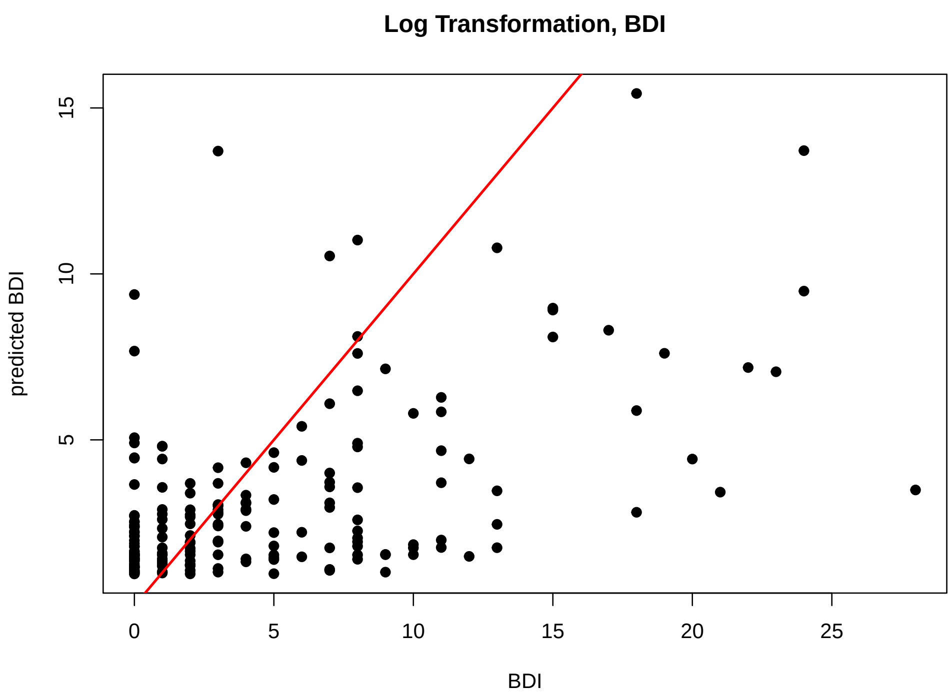
```r
plot(emotion_df[,4], exp(preds)-1, pch=19, main = "Log Transformation, BAI",
     xlab = "BAI", ylab = "predicted BAI")
abline(0,1, col="red", lwd = 2)
```




```r
# log transform, BDI response
log_mod = lm(log(BDI+1) ~ THI_E + TPFQ_E + TFI_E, data = emotion_df)
preds = predict(log_mod, emotion_df[,1:3])
mae(emotion_df[,5], exp(preds)-1)
```

```
## [1] 3.448406
```

```r
plot(emotion_df[,5], exp(preds)-1, pch=19, main = "Log Transformation, BDI",
     xlab = "BDI", ylab = "predicted BDI")
abline(0,1, col="red", lwd = 2)
```



Log transform is slightly better than linear regression.

```
# random forest, BAI
set.seed(1)
rf_mod = randomForest(BAI ~ THI_E + TPFQ_E + TFI_E, data = trn_emotion, mtry = 3)

preds = predict(rf_mod, trn_emotion)
mae(trn_emotion$BAI, preds)
```

```
## [1] 2.133796
```

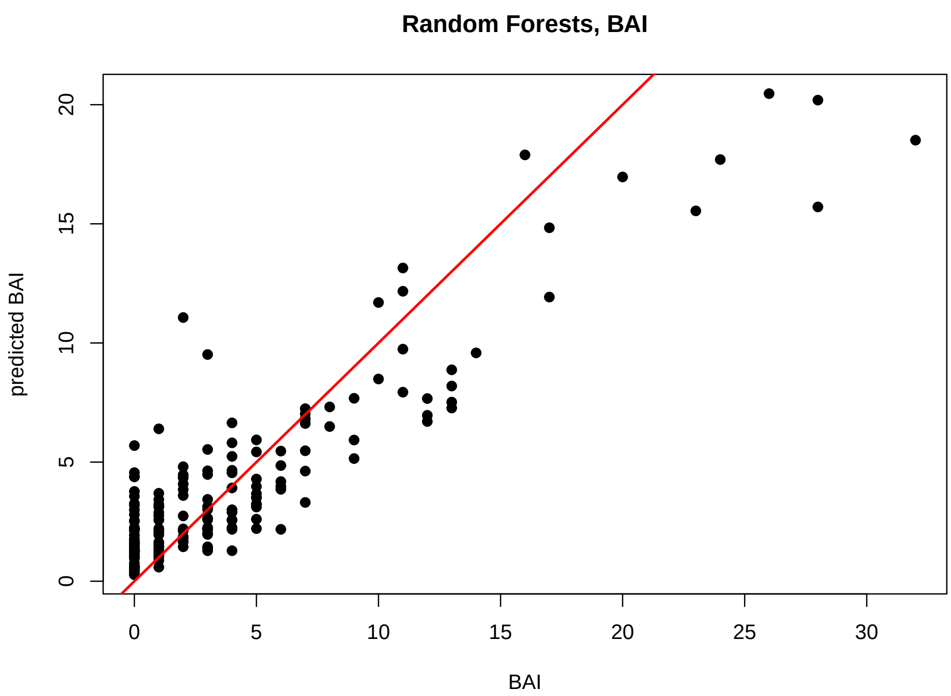
```
preds = predict(rf_mod, tst_emotion)
mae(tst_emotion$BAI, preds)
```

```
## [1] 2.591093
```

```
set.seed(1)
rf_mod = randomForest(BAI ~ THI_E + TPFQ_E + TFI_E, data = emotion_df, mtry = 3)
preds = predict(rf_mod, emotion_df)
mae(emotion_df$BAI, preds)
```

```
## [1] 1.941316
```

```
plot(emotion_df[,4], preds, pch=19, main = "Random Forests, BAI",
     xlab = "BAI", ylab = "predicted BAI")
abline(0,1, col="red", lwd = 2)
```




```r
# random forest, BDI
set.seed(1)
rf_mod = randomForest(BDI ~ THI_E + TPFQ_E + TFI_E, data = trn_emotion, mtry = 3)

preds = predict(rf_mod, trn_emotion)
mae(trn_emotion$BDI, preds)
```

```
## [1] 1.998623
```

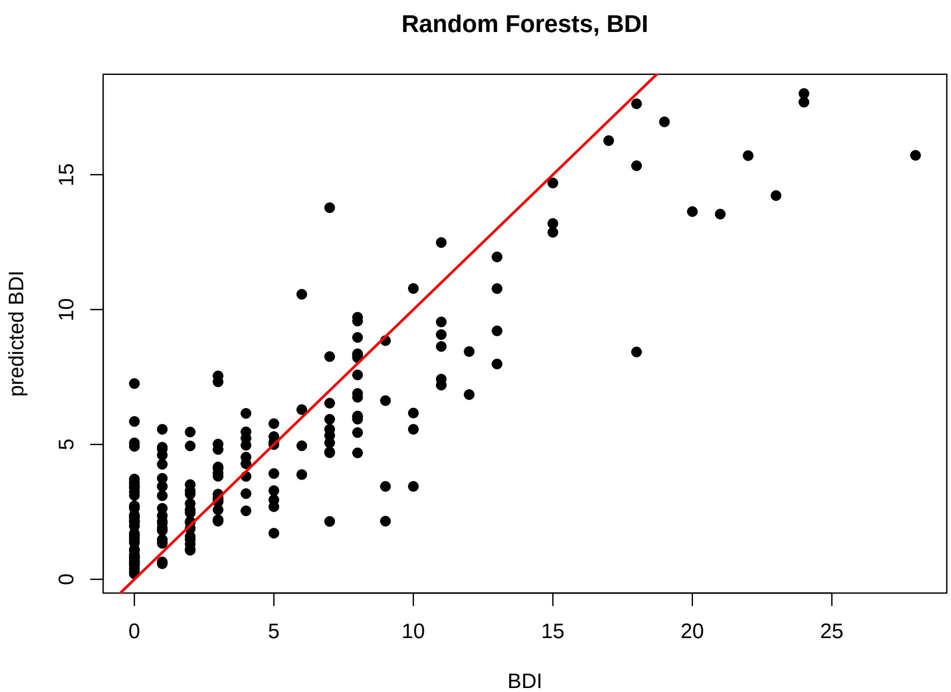
```r
preds = predict(rf_mod, tst_emotion)
mae(tst_emotion$BDI, preds)
```

```
## [1] 4.522016
```

```r
set.seed(1)
rf_mod = randomForest(BDI ~ THI_E + TPFQ_E + TFI_E, data = emotion_df, mtry = 3)
preds = predict(rf_mod, emotion_df)
mae(emotion_df$BDI, preds)
```

```
## [1] 2.102187
```

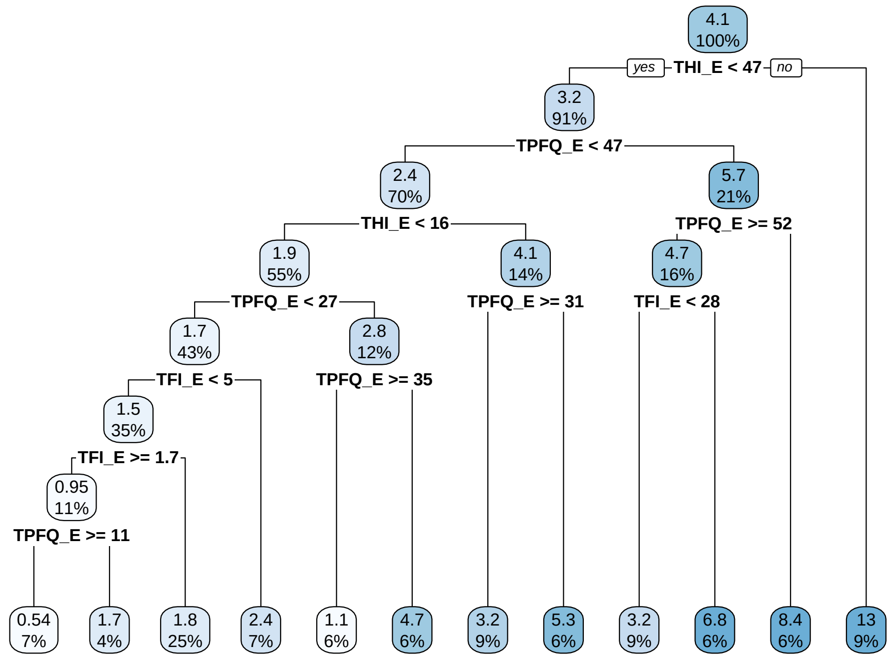
```r
plot(emotion_df[,5], preds, pch=19, main = "Random Forests, BDI",
     xlab = "BDI", ylab = "predicted BDI")
abline(0,1, col="red", lwd = 2)
```



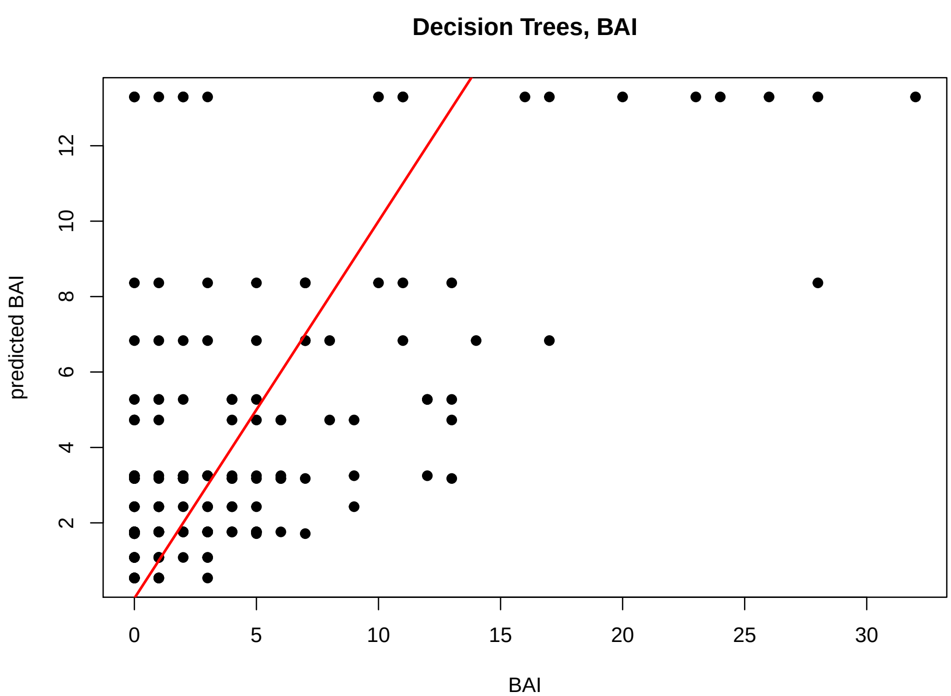

```
# decision tree, BAI
tree_mod = rpart(BAI ~ THI_E + TPFQ_E + TFI_E, data = emotion_df, cp = 0.001)
preds = predict(tree_mod, emotion_df[,1:3])
mae(emotion_df[,4], preds)
```

```
## [1] 3.000346
```

```
rpart.plot(tree_mod)
```



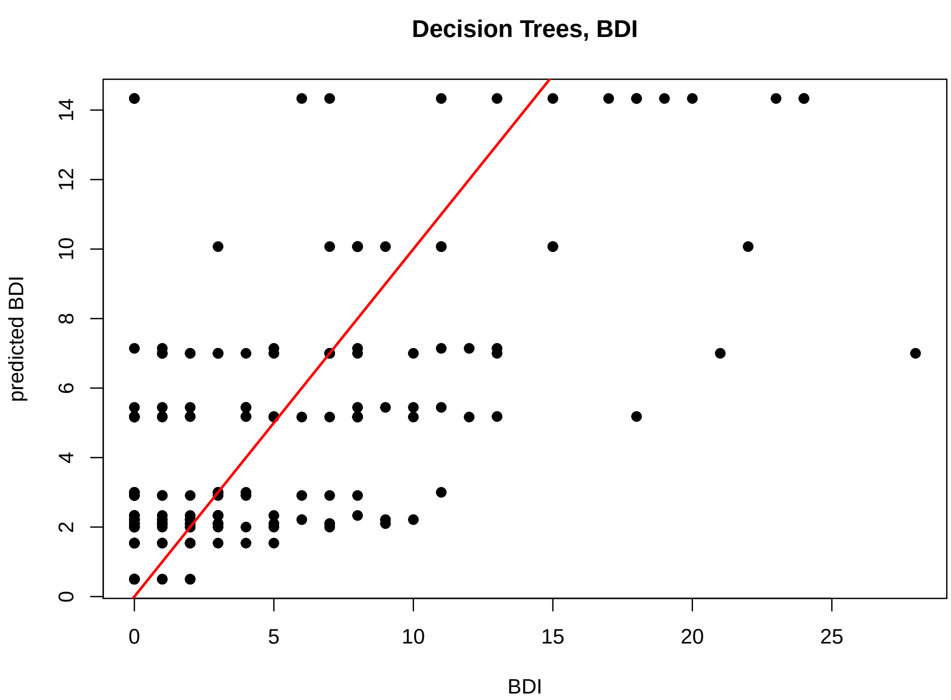

```r
plot(emotion_df[,4], preds, pch=19, main = "Decision Trees, BAI",
     xlab = "BAI", ylab = "predicted BAI")
abline(0,1, col="red", lwd = 2)
```



```r
# decision tree, BDI
tree_mod = rpart(BDI ~ THI_E + TPFQ_E + TFI_E, data = emotion_df, cp = 0.001)
preds = predict(tree_mod, emotion_df[,1:3])
mae(emotion_df[,5], preds)
```

```
## [1] 3.098707
```

```r
plot(emotion_df[,5], preds, pch=19, main = "Decision Trees, BDI",
     xlab = "BDI", ylab = "predicted BDI")
abline(0,1, col="red", lwd = 2)
```




```r
# gbm, BAI
#set.seed(1)
#gbm_mod = gbm(BAI ~ THI_E + TPFQ_E + TFI_E, data = trn_emotion, distribution = "gaussian",
#               n.trees = 10000, interaction.depth = 10, shrinkage = 0.01)
#preds = predict(gbm_mod, trn_emotion)
#mae(trn_emotion$BAI, preds)
```

training MAE 1.99

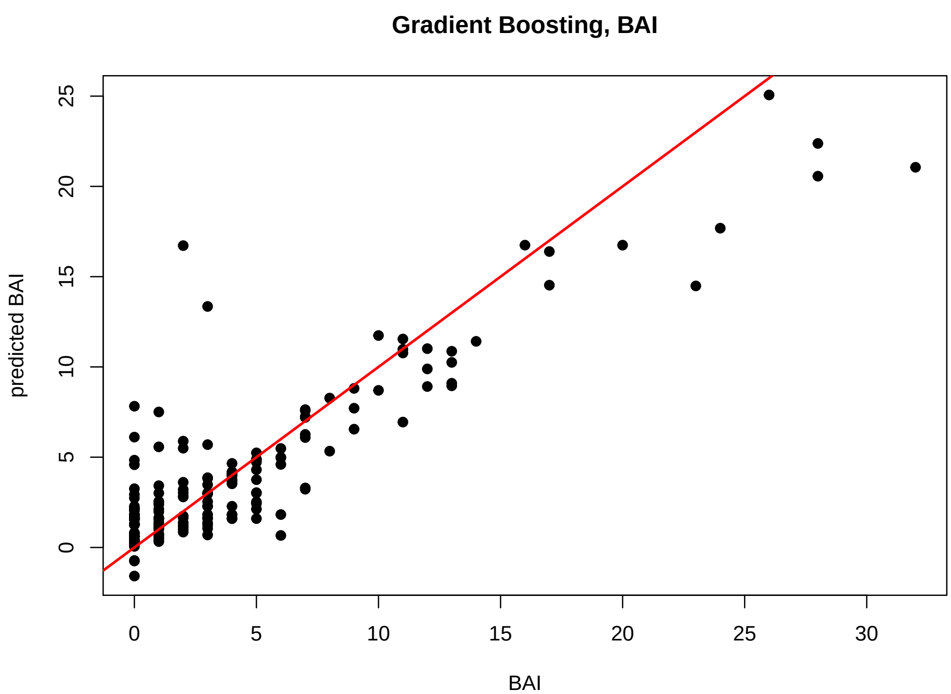
```r
#preds = predict(gbm_mod, tst_emotion)
#mae(tst_emotion$BAI, preds)
```

testing MAE 3.47

```r
set.seed(1)
gbm_mod = gbm(BAI ~ THI_E + TPFQ_E + TFI_E, data = emotion_df, distribution = "gaussian",
              n.trees = 10000, interaction.depth = 10, shrinkage = 0.01)
preds = predict(gbm_mod, emotion_df)
mae(emotion_df$BAI, preds)
```

```
## [1] 1.69088
```

```r
plot(emotion_df[,4], preds, pch=19, main = "Gradient Boosting, BAI",
     xlab = "BAI", ylab = "predicted BAI")
abline(0,1, col="red", lwd = 2)
```




```r
# gbm, BDI
#set.seed(1)
#gbm_mod = gbm(BDI ~ THI_E + TPFQ_E + TFI_E, data = trn_emotion, distribution = "gaussian",
#               n.trees = 10000, interaction.depth = 10, shrinkage = 0.01)
#preds = predict(gbm_mod, trn_emotion)
#mae(trn_emotion$BDI, preds)
```

training MAE 1.638

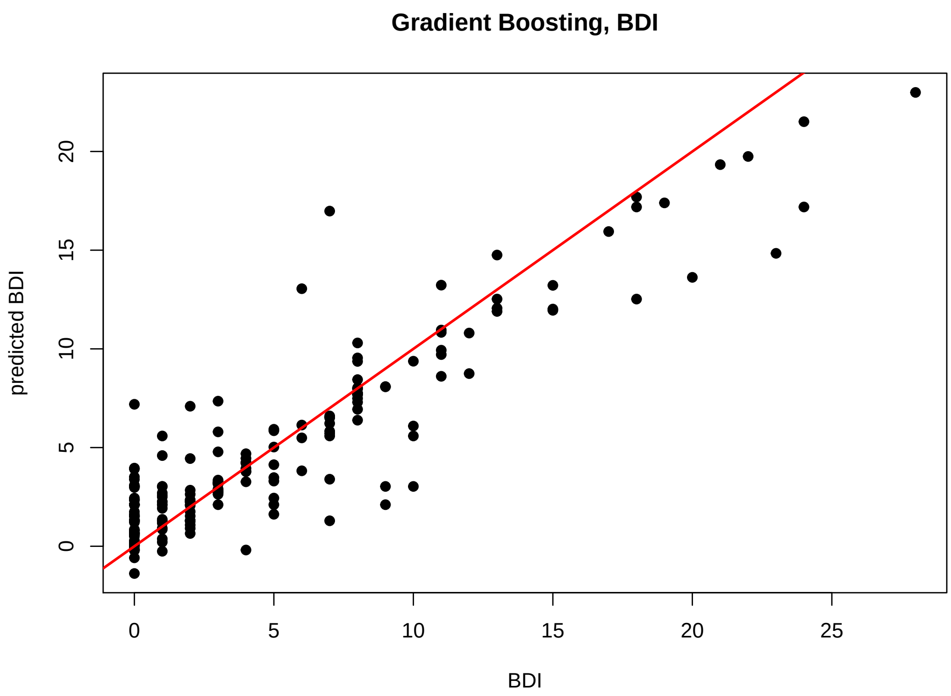
```r
#preds = predict(gbm_mod, tst_emotion)
#mae(tst_emotion$BDI, preds)
```

testing MAE 5.20

```r
set.seed(1)
gbm_mod = gbm(BDI ~ THI_E + TPFQ_E + TFI_E, data = emotion_df, distribution = "gaussian",
              n.trees = 10000, interaction.depth = 10, shrinkage = 0.01)
preds = predict(gbm_mod, emotion_df)
mae(emotion_df$BDI, preds)
```

```
## [1] 1.581983
```

```r
plot(emotion_df[,5], preds, pch=19, main = "Gradient Boosting, BDI",
     xlab = "BDI", ylab = "predicted BDI")
abline(0,1, col="red", lwd = 2)
```




```r
# transform BAI and BDI to binary varibles
emotion_df$BAI_bin = as.numeric(emotion_df[,4]<=2)
emotion_df$BDI_bin = as.numeric(emotion_df[,5]<=2)
```

```r
# logistic regression, BAI binary
log_mod = glm(BAI_bin ~ THI_E + TPFQ_E + TFI_E, data = emotion_df, family = "binomial")
preds = ifelse(predict(log_mod, emotion_df[,1:3], type = "response")>0.5, 1, 0)
mean(emotion_df[,6]==preds)
```

```
## [1] 0.631016
```

```r
mean(emotion_df$BAI_bin)
```

```
## [1] 0.540107
```

Accuracy 0.63, NIR 0.54. Terrible.

```
# logistic regression, BDI binary
log_mod_2 = glm(BDI_bin ~ THI_E + TPFQ_E + TFI_E, data = emotion_df, family = "binomial")
preds = ifelse(predict(log_mod_2, emotion_df[,1:3], type = "response")>0.5, 1, 0)
mean(emotion_df[,7]==preds)
```

```
## [1] 0.6791444
```

```
1-mean(emotion_df$BDI_bin)
```

```
## [1] 0.513369
```

Accuracy 0.68, NIR 0.51. Terrible.

```
# combine BAI and BDI
emotion_df$BAI.BDI.bin = emotion_df$BAI_bin * emotion_df$BDI_bin

# logistic regression, combined
log_mod_3 = glm(BAI.BDI.bin ~ THI_E + TPFQ_E + TFI_E, data = emotion_df, family = "binomial")
preds = ifelse(predict(log_mod_3, emotion_df[,1:3], type = "response")>0.5, 1, 0)
mean(emotion_df$BAI.BDI.bin==preds)
```

```
## [1] 0.657754
```

```
1-mean(emotion_df$BAI.BDI.bin)
```

```
## [1] 0.6096257
```

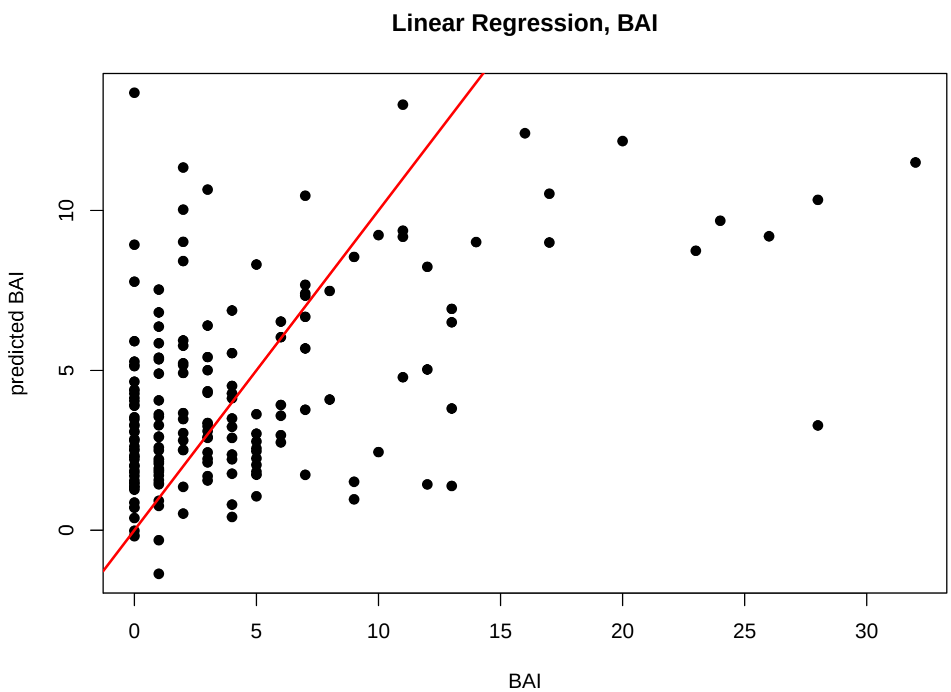
Accuracy 0.66, NIR 0.61. Terrible again.

```
# Since TFI is the best, try to use its subscale to predict
# extract TFI columns
TFI_data = data.matrix(data[,c(16,17,18,19,20,21,22,23,24,25)])
TFI_data = TFI_data[complete.cases(TFI_data),]
colnames(TFI_data) = c("Intrusive","Control","Cognition","Sleep","Auditory","Relax",
                       "Emotion","Quality","BAI","BDI")
TFI_df = data.frame(TFI_data)
trn_data = TFI_df[-tst_idx,]
tst_data = TFI_df[tst_idx,]
```

```
# linear regression, BAI response
lm_mod = lm(BAI ~ Intrusive + Control + Cognition + Sleep + Auditory + Relax +
                  Emotion + Quality, data = TFI_df)
preds = predict(lm_mod, TFI_df)
mae(TFI_df$BAI, preds)
```

```
## [1] 3.311383
```

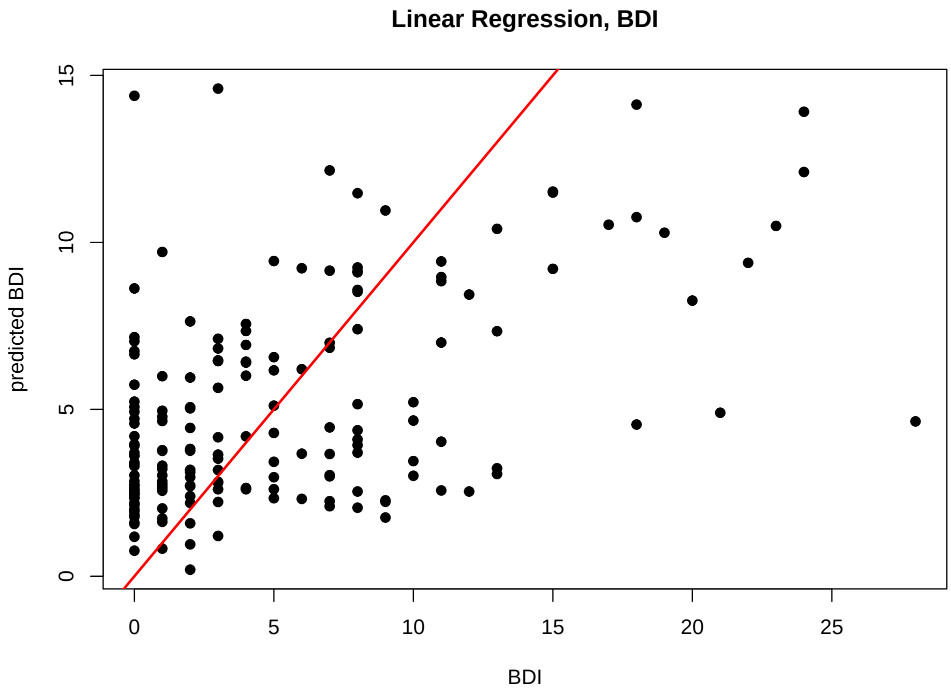
```r
plot(TFI_df$BAI, preds, pch=19, main = "Linear Regression, BAI",
     xlab = "BAI", ylab = "predicted BAI")
abline(0,1, col="red", lwd = 2)
```




```r
# linear regression, BDI response
lm_mod_2 = lm(BDI ~ Intrusive + Control + Cognition + Sleep + Auditory + Relax +
                     Emotion + Quality, data = TFI_df)
preds = predict(lm_mod_2, TFI_df)
mae(TFI_df$BDI, preds)
```

```
## [1] 3.644403
```

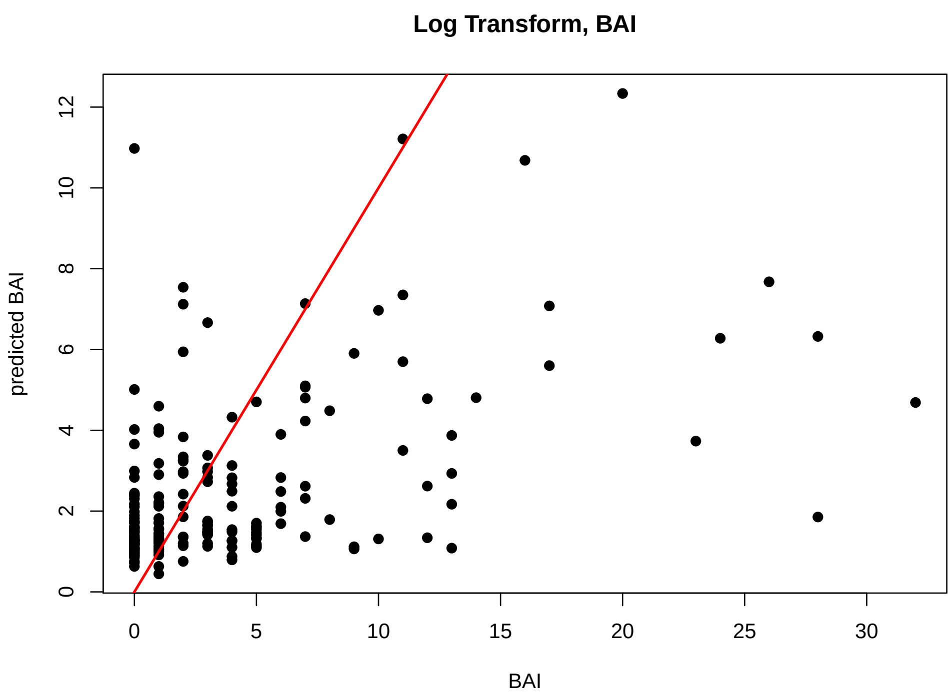
```r
plot(TFI_df$BDI, preds, pch=19, main = "Linear Regression, BDI",
     xlab = "BDI", ylab = "predicted BDI")
abline(0,1, col="red", lwd = 2)
```




```
# log transform, BAI response
log_mod = lm(log(BAI+1) ~ Intrusive + Control + Cognition + Sleep + Auditory + Relax +
                          Emotion + Quality, data = TFI_df)
preds = predict(log_mod, TFI_df)
mae(TFI_df$BAI, exp(preds)-1)
```

```
## [1] 3.116057
```

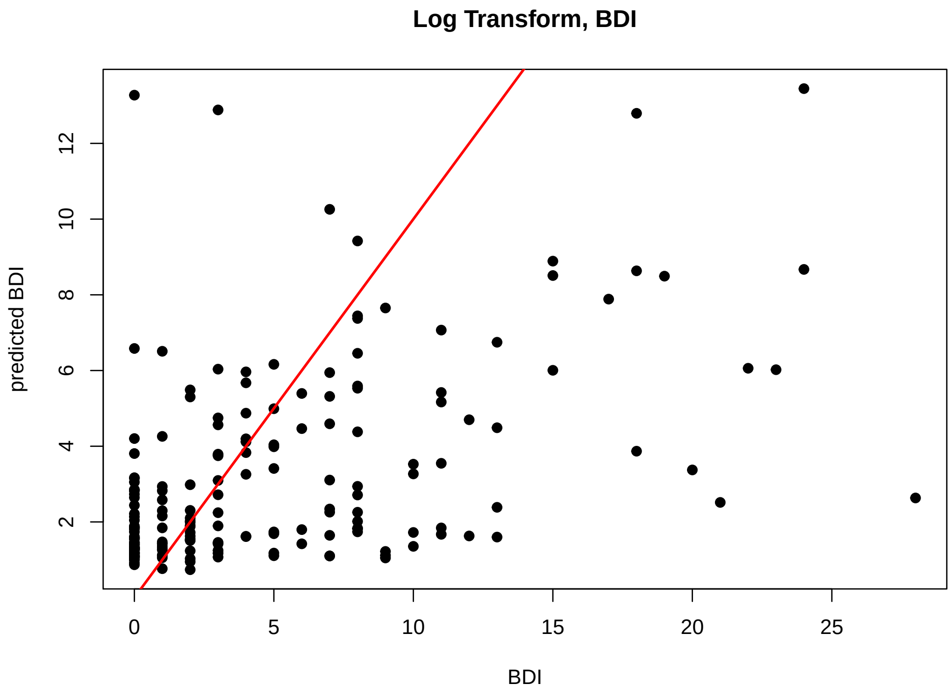
```
plot(TFI_df$BAI, exp(preds)-1, pch=19, main = "Log Transform, BAI",
     xlab = "BAI", ylab = "predicted BAI")
abline(0,1, col="red", lwd = 2)
```




```r
# log transform, BDI response
log_mod = lm(log(BDI+1) ~ Intrusive + Control + Cognition + Sleep + Auditory + Relax +
                          Emotion + Quality, data = TFI_df)
preds = predict(log_mod, TFI_df)
mae(TFI_df$BDI, exp(preds)-1)
```

```
## [1] 3.448823
```

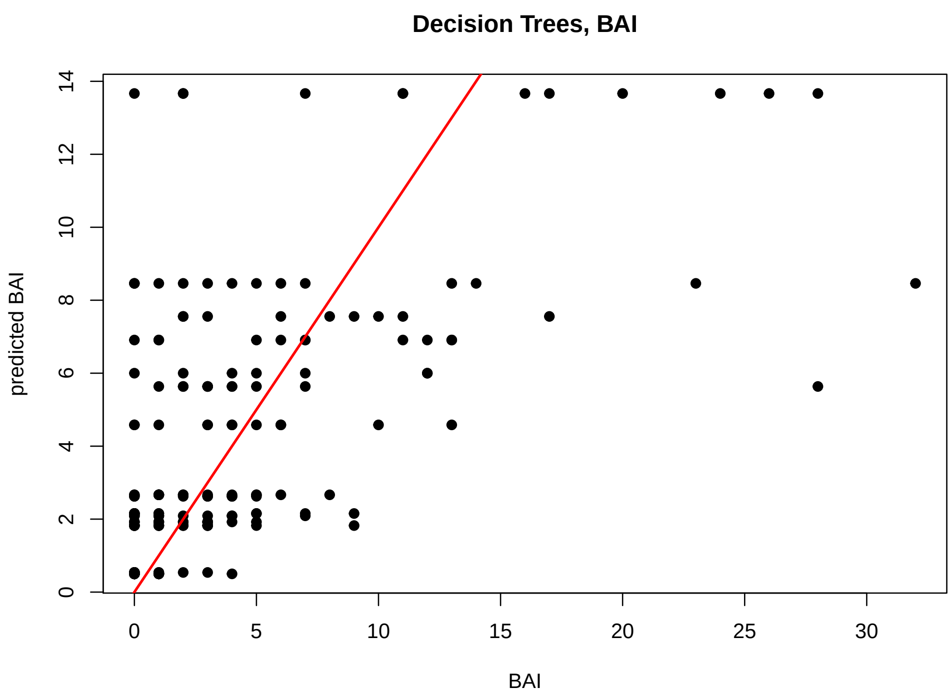
```r
plot(TFI_df$BDI, exp(preds)-1, pch=19, main = "Log Transform, BDI",
     xlab = "BDI", ylab = "predicted BDI")
abline(0,1, col="red", lwd = 2)
```




```
# decision tree, BAI
tree_mod = rpart(BAI ~ Intrusive + Control + Cognition + Sleep + Auditory + Relax +
                       Emotion + Quality, data = TFI_df, cp = 0.001)
preds = predict(tree_mod, TFI_df)
mae(TFI_df$BAI, preds)
```

```
## [1] 2.911744
```

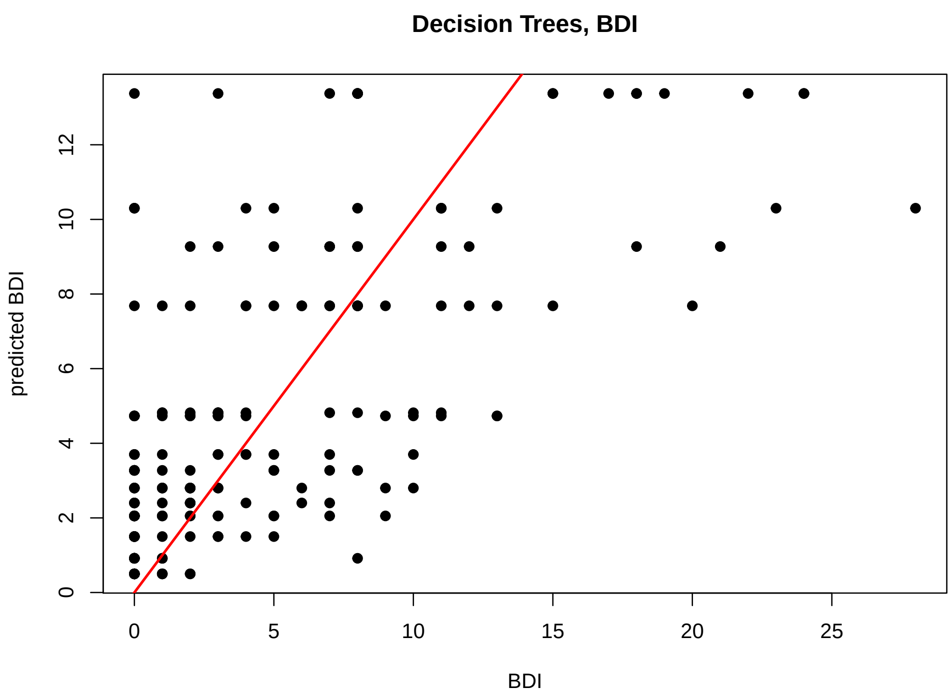
```
plot(TFI_df$BAI, preds, pch=19, main = "Decision Trees, BAI",
     xlab = "BAI", ylab = "predicted BAI")
abline(0,1, col="red", lwd = 2)
```




```
# decision tree, BDI
tree_mod = rpart(BDI ~ Intrusive + Control + Cognition + Sleep + Auditory + Relax +
                        Emotion + Quality, data = TFI_df, cp = 0.001)
preds = predict(tree_mod, TFI_df)
mae(TFI_df$BDI, preds)
```

```
## [1] 3.116478
```

```
plot(TFI_df$BDI, preds, pch=19, main = "Decision Trees, BDI",
     xlab = "BDI", ylab = "predicted BDI")
abline(0,1, col="red", lwd = 2)
```




```
# random forest, BAI
set.seed(1)
rf_mod = randomForest(BAI ~ Intrusive + Control + Cognition + Sleep + Auditory + Relax +
                            Emotion + Quality, data = trn_data, mtry = 8)

preds = predict(rf_mod, trn_data)
mae(trn_data$BAI, preds)
```

```
## [1] 1.740493
```

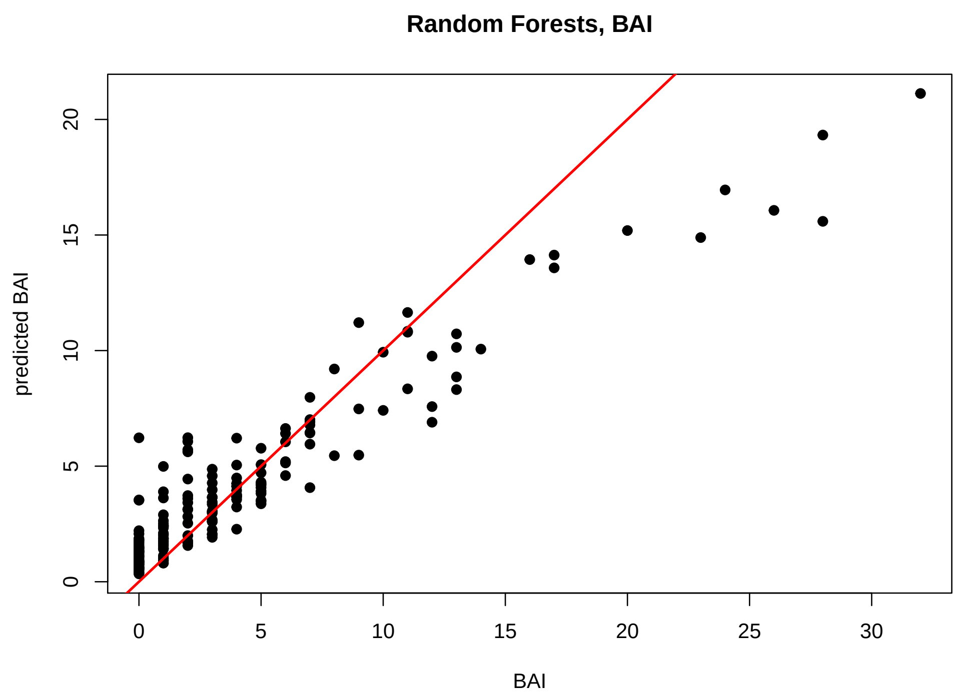
```
preds = predict(rf_mod, tst_data)
mae(tst_data$BAI, preds)
```

```
## [1] 2.502299
```

```
set.seed(1)
rf_mod = randomForest(BAI ~ Intrusive + Control + Cognition + Sleep + Auditory + Relax +
                            Emotion + Quality, data = TFI_df, mtry = 8)
preds = predict(rf_mod, TFI_df)
mae(TFI_df$BAI, preds)
```

```
## [1] 1.530498
```

```
plot(TFI_df$BAI, preds, pch=19, main = "Random Forests, BAI",
     xlab = "BAI", ylab = "predicted BAI")
abline(0,1, col="red", lwd = 2)
```




```r
# random forest, BDI
set.seed(1)
rf_mod = randomForest(BDI ~ Intrusive + Control + Cognition + Sleep + Auditory + Relax +
                          Emotion + Quality, data = trn_data, mtry = 8)

preds = predict(rf_mod, trn_data)
mae(trn_data$BDI, preds)
```

```
## [1] 1.688217
```

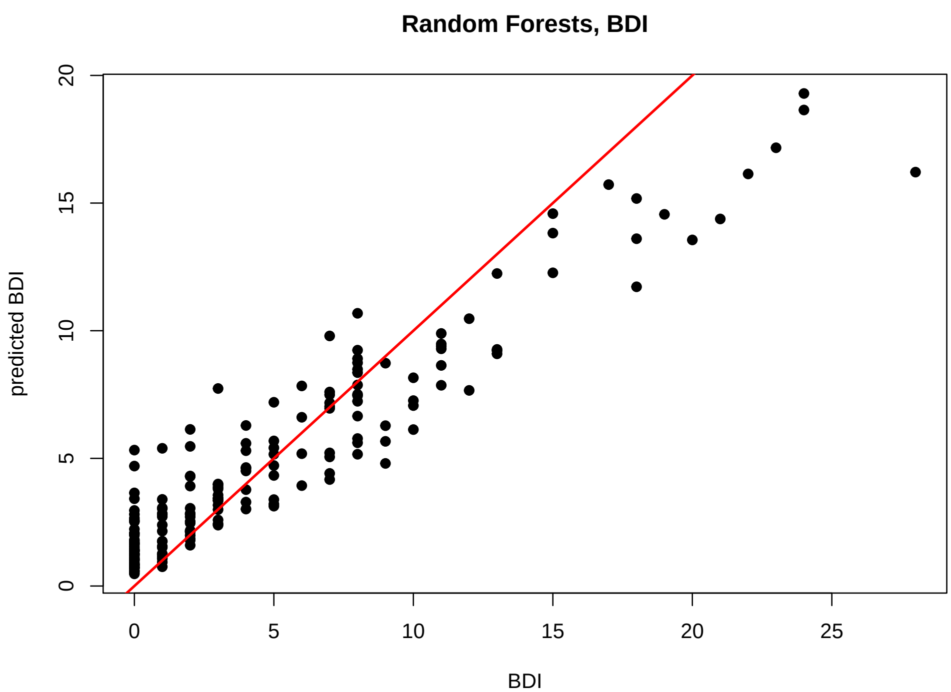
```r
preds = predict(rf_mod, tst_data)
mae(tst_data$BDI, preds)
```

```
## [1] 4.128537
```

```r
set.seed(1)
rf_mod = randomForest(BDI ~ Intrusive + Control + Cognition + Sleep + Auditory + Relax +
                          Emotion + Quality, data = TFI_df, mtry = 8)

preds = predict(rf_mod, TFI_df)
mae(TFI_df$BDI, preds)
```

```
## [1] 1.682938
```

```r
plot(TFI_df$BDI, preds, pch=19, main = "Random Forests, BDI",
     xlab = "BDI", ylab = "predicted BDI")
abline(0,1, col="red", lwd = 2)
```




```
fitControl = trainControl(## 5-fold CV
                          method = "repeatedcv",
                          number = 5,
                          ## repeated 5 times
                          repeats = 5)
gbmGrid <-  expand.grid(interaction.depth = c(2,4,6,8,10),
                        n.trees = c(100, 200, 500, 1000),
                        shrinkage = 0.01,
                        n.minobsinnode = c(2, 4))
set.seed(1)
gbm_mod = train(BAI ~ Intrusive + Control + Cognition + Sleep + Auditory + Relax +
                 Emotion + Quality, data = TFI_df,
                 method = "gbm",
                 trControl = fitControl,
                 ## This last option is actually one
                 ## for gbm() that passes through
                 verbose = FALSE, tuneGrid = gbmGrid)
gbm_mod
```

```
## Stochastic Gradient Boosting
##
## 187 samples
##   8 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 150, 150, 149, 149, 150, 149, ...
## Resampling results across tuning parameters:
##
```

```
##   interaction.depth  n.minobsinnode  n.trees  RMSE      Rsquared    MAE
##    2                 2                 100    5.353284  0.11814496  3.628578
##    2                 2                 200    5.410310  0.11499381  3.599863
##    2                 2                 500    5.619655  0.10636210  3.679636
##    2                 2                1000    5.774907  0.09930798  3.781815
##    2                 4                 100    5.285761  0.14790190  3.585796
##    2                 4                 200    5.310183  0.14087333  3.554964
##    2                 4                 500    5.477476  0.12772431  3.628747
##    2                 4                1000    5.641022  0.11919569  3.740983
##    4                 2                 100    5.319884  0.13164989  3.588249
##    4                 2                 200    5.419543  0.12258762  3.590791
##    4                 2                 500    5.653891  0.10878358  3.714440
##    4                 2                1000    5.817251  0.09962226  3.826697
##    4                 4                 100    5.287518  0.14352185  3.583585
##    4                 4                 200    5.331068  0.14037887  3.566654
##    4                 4                 500    5.539093  0.12837434  3.685908
##    4                 4                1000    5.734904  0.11848842  3.828062
##    6                 2                 100    5.350536  0.12340429  3.625685
##    6                 2                 200    5.461402  0.11528602  3.635278
##    6                 2                 500    5.724763  0.10374254  3.781429
##    6                 2                1000    5.887205  0.09263526  3.898402
##    6                 4                 100    5.299996  0.13897987  3.610371
##    6                 4                 200    5.354100  0.13599761  3.603161
##    6                 4                 500    5.577609  0.12372076  3.731355
##    6                 4                1000    5.779996  0.10878118  3.884062
##    8                 2                 100    5.368570  0.11438768  3.618653
##    8                 2                 200    5.479740  0.11469618  3.643782
##    8                 2                 500    5.763898  0.10127379  3.778479
##    8                 2                1000    5.932078  0.08663511  3.906657
##    8                 4                 100    5.304811  0.13875756  3.598850
##    8                 4                 200    5.382029  0.13289422  3.598001
##    8                 4                 500    5.607410  0.12152263  3.739218
##    8                 4                1000    5.822341  0.10455688  3.912235
##   10                 2                 100    5.373381  0.11756429  3.629991
##   10                 2                 200    5.500742  0.11295758  3.664044
##   10                 2                 500    5.800391  0.09689356  3.815396
##   10                 2                1000    5.982082  0.08139489  3.940073
##   10                 4                 100    5.311352  0.13834983  3.593267
##   10                 4                 200    5.368216  0.13789046  3.599079
##   10                 4                 500    5.624896  0.11771116  3.777689
##   10                 4                1000    5.835251  0.10158563  3.933549
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.01
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were n.trees = 100, interaction.depth = 2,
##  shrinkage = 0.01 and n.minobsinnode = 4.
```

```r
# gbm, BAI
set.seed(1)
gbm_mod = gbm(BAI ~ Intrusive + Control + Cognition + Sleep + Auditory + Relax +
              Emotion + Quality, data = trn_data, distribution = "gaussian",
              n.trees = 10000, interaction.depth = 10, shrinkage = 0.01)

preds = predict(gbm_mod, trn_data)
```

```r
mae(trn_data$BAI, preds)
```

```
## [1] 0.3623556
```

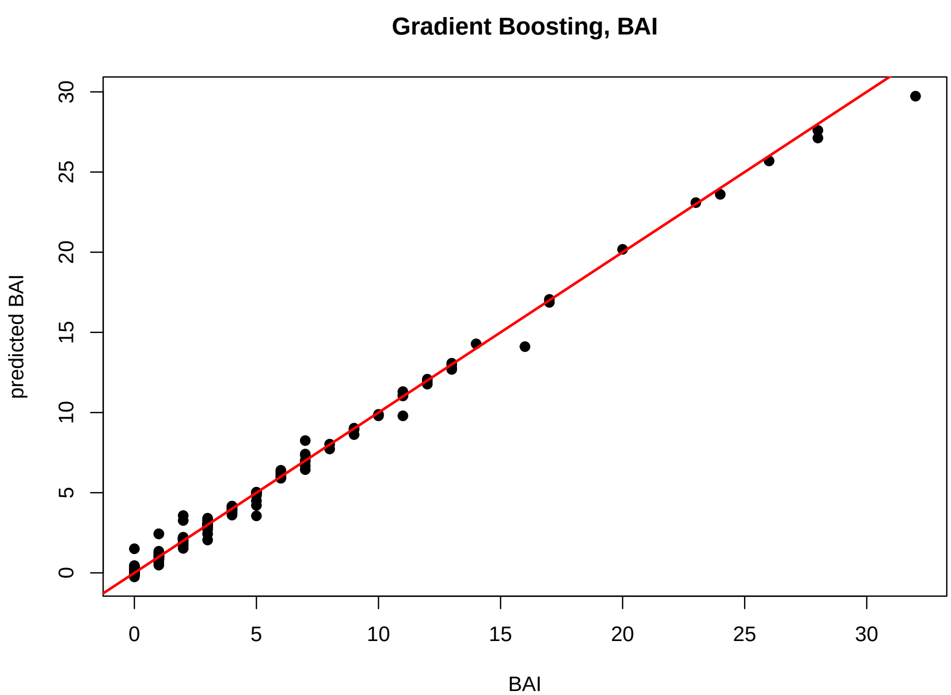
```r
preds = predict(gbm_mod, tst_data)
mae(tst_data$BAI, preds)
```

```
## [1] 3.476004
```

```r
set.seed(1)
gbm_mod = gbm(BAI ~ Intrusive + Control + Cognition + Sleep + Auditory + Relax +
              Emotion + Quality, data = TFI_df, distribution = "gaussian",
              n.trees = 10000, interaction.depth = 10, shrinkage = 0.01)

preds = predict(gbm_mod, TFI_df)
mae(TFI_df$BAI, preds)
```

```
## [1] 0.2338378
```

```r
plot(TFI_df$BAI, preds, pch=19, main = "Gradient Boosting, BAI",
     xlab = "BAI", ylab = "predicted BAI")
abline(0,1, col="red", lwd = 2)
```




```r
# gbm, BDI
set.seed(1)
gbm_mod = gbm(BDI ~ Intrusive + Control + Cognition + Sleep + Auditory + Relax +
```

```r
               Emotion + Quality, data = trn_data, distribution = "gaussian",
               n.trees = 10000, interaction.depth = 10, shrinkage = 0.01)

preds = predict(gbm_mod, trn_data)
mae(trn_data$BDI, preds)
```

```
## [1] 0.3122058
```

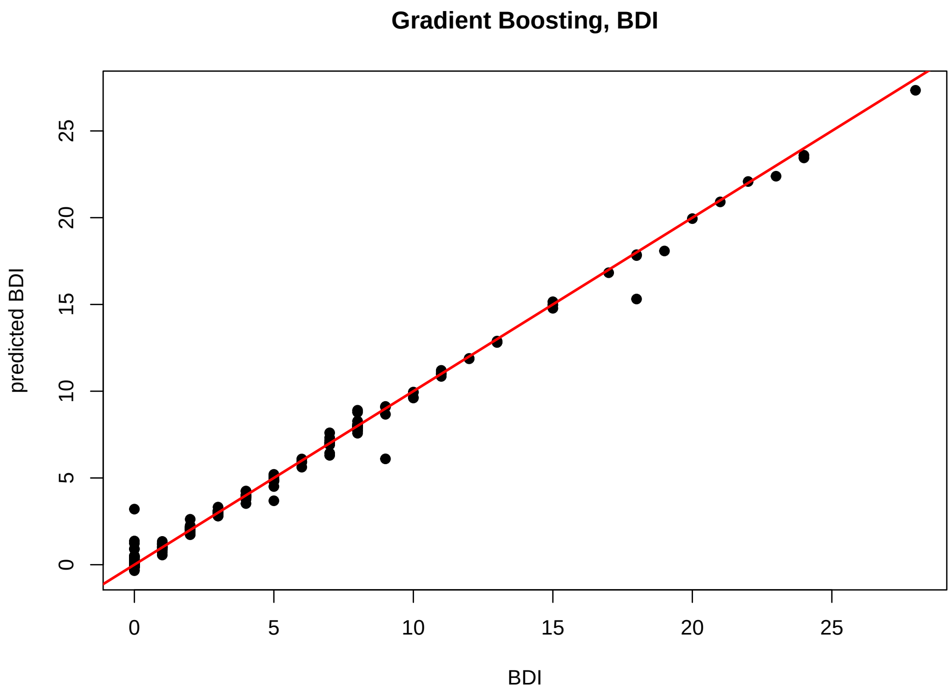
```r
preds = predict(gbm_mod, tst_data)
mae(tst_data$BDI, preds)
```

```
## [1] 5.063058
```

```r
set.seed(1)
gbm_mod = gbm(BDI ~ Intrusive + Control + Cognition + Sleep + Auditory + Relax +
               Emotion + Quality, data = TFI_df, distribution = "gaussian",
               n.trees = 10000, interaction.depth = 10, shrinkage = 0.01)

preds = predict(gbm_mod, TFI_df)
mae(TFI_df$BDI, preds)
```

```
## [1] 0.2513759
```

```r
plot(TFI_df$BDI, preds, pch=19, main = "Gradient Boosting, BDI",
     xlab = "BDI", ylab = "predicted BDI")
abline(0,1, col="red", lwd = 2)
```



```r
# transform BAI and BDI to binary varibles
TFI_df$BAI_bin = as.numeric(TFI_df$BAI<=2)
TFI_df$BDI_bin = as.numeric(TFI_df$BDI<=2)
```

```r
# logistic regression, BAI binary
log_mod = glm(BAI_bin ~ Intrusive + Control + Cognition + Sleep + Auditory + Relax +
                Emotion + Quality, data = TFI_df, family = "binomial")
preds = ifelse(predict(log_mod, TFI_df[,1:8], type = "response")>0.5, 1, 0)
mean(TFI_df$BAI_bin==preds)
```

```
## [1] 0.6363636
```

```r
mean(TFI_df$BAI_bin)
```

```
## [1] 0.540107
```

Accuracy 0.64, NIR 0.54. Terrible.

```r
# logistic regression, BDI binary
log_mod_2 = glm(BDI_bin ~ Intrusive + Control + Cognition + Sleep + Auditory + Relax +
                  Emotion + Quality, data = TFI_df, family = "binomial")
preds = ifelse(predict(log_mod_2, TFI_df[,1:8], type = "response")>0.5, 1, 0)
mean(TFI_df$BDI_bin==preds)
```

```
## [1] 0.7005348
```

```r
1-mean(TFI_df$BDI_bin)
```

```
## [1] 0.513369
```

Accuracy 0.70, NIR 0.51. Terrible.

```r
# combine BAI and BDI
TFI_df$BAI.BDI.bin = TFI_df$BAI_bin * TFI_df$BDI_bin

# logistic regression, combined
log_mod_3 = glm(BAI.BDI.bin ~ Intrusive + Control + Cognition + Sleep + Auditory + Relax +
                  Emotion + Quality, data = TFI_df, family = "binomial")
preds = ifelse(predict(log_mod_3, TFI_df[,1:8], type = "response")>0.5, 1, 0)
mean(TFI_df$BAI.BDI.bin==preds)
```

```
## [1] 0.7326203
```

```r
mean(TFI_df$BAI_bin)
```

```
## [1] 0.540107
```

Accuracy 0.73, NIR 0.54. **Still not good at all, but the best so far in our first trial.**

Improvements:

- 1. classification, SMOTE upsample (done)

```r
######### pooled emotional data
library(smotefamily)
## reclassification by BAI and BDI depression criteria
emotion_df$BAI_bin <- emotion_df$BAI < 21
emotion_df$BDI_bin <- emotion_df$BDI < 16
emotion_df$BAI.BDI.bin <- emotion_df$BAI_bin*emotion_df$BDI_bin
# check data balance
mean(emotion_df$BAI.BDI.bin)
```

```
## [1] 0.9304813
```

```r
## using SMOTE to correct unbalanced data(response)
gen_data <- SMOTE(emotion_df[,1:3],emotion_df[,8])
test_data <- gen_data$data
# check balance
mean(as.numeric(test_data$class))
```

```
## [1] 0.5072886
```

```r
## logistic regression
log_mod = glm(as.factor(class) ~ THI_E + TPFQ_E + TFI_E, data = test_data, family = "binomial")
preds = ifelse(predict(log_mod, test_data[,1:3], type = "response")>0.5, 1, 0)
mean(test_data$class==preds)
```

```
## [1] 0.8104956
```

```r
preds = ifelse(predict(log_mod, emotion_df[,1:3], type = "response")>0.5, 1, 0)
mean(emotion_df$BAI.BDI.bin==preds)
```

```
## [1] 0.7967914
```

```r
######### TFI data
## reclassification by BAI and BDI depression criteria
TFI_df$BAI_bin <- TFI_df$BAI < 21
TFI_df$BDI_bin <- TFI_df$BDI < 16
TFI_df$BAI.BDI.bin <- TFI_df$BAI_bin*TFI_df$BDI_bin
# check data balance
mean(TFI_df$BAI.BDI.bin)
```

```
## [1] 0.9304813
```

```r
## using SMOTE to correct unbalanced data(response)
gen_data <- SMOTE(TFI_df[,1:8],TFI_df[,13])
test_data <- gen_data$data
# check balance
mean(as.numeric(test_data$class))
```

```
## [1] 0.5072886
```

```r
## logistic regression
cat(colnames(test_data))
```

```
## Intrusive Control Cognition Sleep Auditory Relax Emotion Quality class
```

```r
log_mod = glm(as.factor(class) ~ Intrusive +Control +Cognition +Sleep +Auditory +Relax +Emotion +Quality
preds = ifelse(predict(log_mod, test_data[,1:8], type = "response")>0.5, 1, 0)
mean(test_data$class==preds)
```

```
## [1] 0.7784257
```

```r
preds = ifelse(predict(log_mod, TFI_df[,1:8], type = "response")>0.5, 1, 0)
mean(TFI_df$BAI.BDI.bin==preds)
```

```
## [1] 0.7914439
```

```r
table(TFI_df$BAI.BDI.bin,preds)
```

```
##    preds
##       0   1
##   0   9   4
##   1  35 139
```

- 2. BDI Q17, BAI Q6, Q16, Q19 (done)

```r
merged = read.csv("merged_copy.csv")
merged = merged[,4:114]
merged = merged[complete.cases(merged),]

df = data.frame(THI_E = rowSums(merged[,c(3,6,10,16,17,21,22,25)])/32*100,
                TPFQ_E = rowSums(merged[,c(53,58,63,65,68)])/5,
                TFI_I = rowSums(merged[,c(26,27,28)])/3*10,
                TFI_SC = rowSums(merged[,c(29,30,31)])/3*10,
                TFI_C = rowSums(merged[,c(32,33,34)])/3*10,
                TFI_SL = rowSums(merged[,c(35,36,37)])/3*10,
                TFI_A = rowSums(merged[,c(38,39,40)])/3*10,
                TFI_R = rowSums(merged[,c(41,42,43)])/3*10,
                TFI_Q = rowSums(merged[,c(44,45,46,47)])/3*10,
                TFI_E = rowSums(merged[,c(48,49,50)])/3*10,
                BDI_Q17 = ifelse(merged[,86] > 0, 1, 0),
                BAI_Q6  = ifelse(merged[,96] > 0, 1, 0),
                BAI_Q11 = ifelse(merged[,101] > 0, 1, 0),
                BAI_Q16 = ifelse(merged[,106] > 0, 1, 0),
                BAI_Q18 = ifelse(merged[,108] > 0, 1, 0),
                BAI_Q19 = ifelse(merged[,109] > 0, 1, 0),
                BDI_total = rowSums(merged[,71:90]),
                BAI_total = rowSums(merged[,91:111]))
```

```r
table(df$BAI_Q19)
```

```
##
##   0   1
## 171   3
```

```r
# logistic regression, BAI_Q19, ROSE upsampling, Emotional subscale predictors
set.seed(1)
rose_df = ROSE(BAI_Q19 ~ THI_E + TPFQ_E + TFI_E, data  = df)$data

BAI_Q19_mod = glm(BAI_Q19 ~ ., data  = rose_df, family = "binomial")
preds = ifelse(predict(BAI_Q19_mod, rose_df[,1:3], type = "response")>0.5, 1, 0)
mean(rose_df$BAI_Q19==preds)
```

```
## [1] 0.908046
```

```r
1-mean(rose_df$BAI_Q19)
```

```
## [1] 0.5114943
```

```r
# logistic regression, BAI_Q19, ROSE upsampling, TFI predictors
set.seed(1)
rose_df = ROSE(BAI_Q19 ~ TFI_I + TFI_SC + TFI_C + TFI_SL + TFI_A + TFI_R + TFI_Q + TFI_E, data  = df)$da

BAI_Q19_mod = glm(BAI_Q19 ~ ., data  = rose_df, family = "binomial")
preds = ifelse(predict(BAI_Q19_mod, rose_df[,1:8], type = "response")>0.5, 1, 0)
mean(rose_df$BAI_Q19==preds)
```

```
## [1] 0.9770115
```

```r
1-mean(rose_df$BAI_Q19)
```

```
## [1] 0.5114943
```

BAI Q19 is significant for both emotional subscale predictors and TFI predictors.

```r
# logistic regression, BAI_Q11, ROSE upsampling, TFI predictors
set.seed(1)
rose_df = ROSE(BAI_Q11 ~ TFI_I + TFI_SC + TFI_C + TFI_SL + TFI_A + TFI_R + TFI_Q + TFI_E, data  = df)$da

BAI_Q11_mod = glm(BAI_Q11 ~ ., data  = rose_df, family = "binomial")
preds = ifelse(predict(BAI_Q11_mod, rose_df[,1:8], type = "response")>0.5, 1, 0)
mean(rose_df$BAI_Q11==preds)
```

```
## [1] 0.7931034
```

```r
1-mean(rose_df$BAI_Q11)
```

```
## [1] 0.5114943
```

BAI Q11 is significant for TFI predictors.

Other results:

- BAI Q19 (0.91, 0.98)
- BAI Q18 (0.54, 0.65)
- BAI Q16 (0.67, 0.68)
- BAI Q11 (0.65, 0.79)
- BAI Q6 (0.64, 0.63)
- BDI Q17 (0.66, 0.63)