Untitled

Project Group 9

November 30, 2015

Trees

Read & Clean

```
# Read the data from CSV
yelp = read.csv("YelpUsers-V2.csv")

# Remove less useful columns from the dataset (country, name, Id)
yelp = yelp[, -c(1:3)]

# Check the data for missing values
sum(is.na(yelp))
```

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

Extract only Elite users

```
# Remove non-Elite members from dataset.
yelpElite = yelp[(yelp$EliteYearCount > 0), ]
```

Count years to become elite

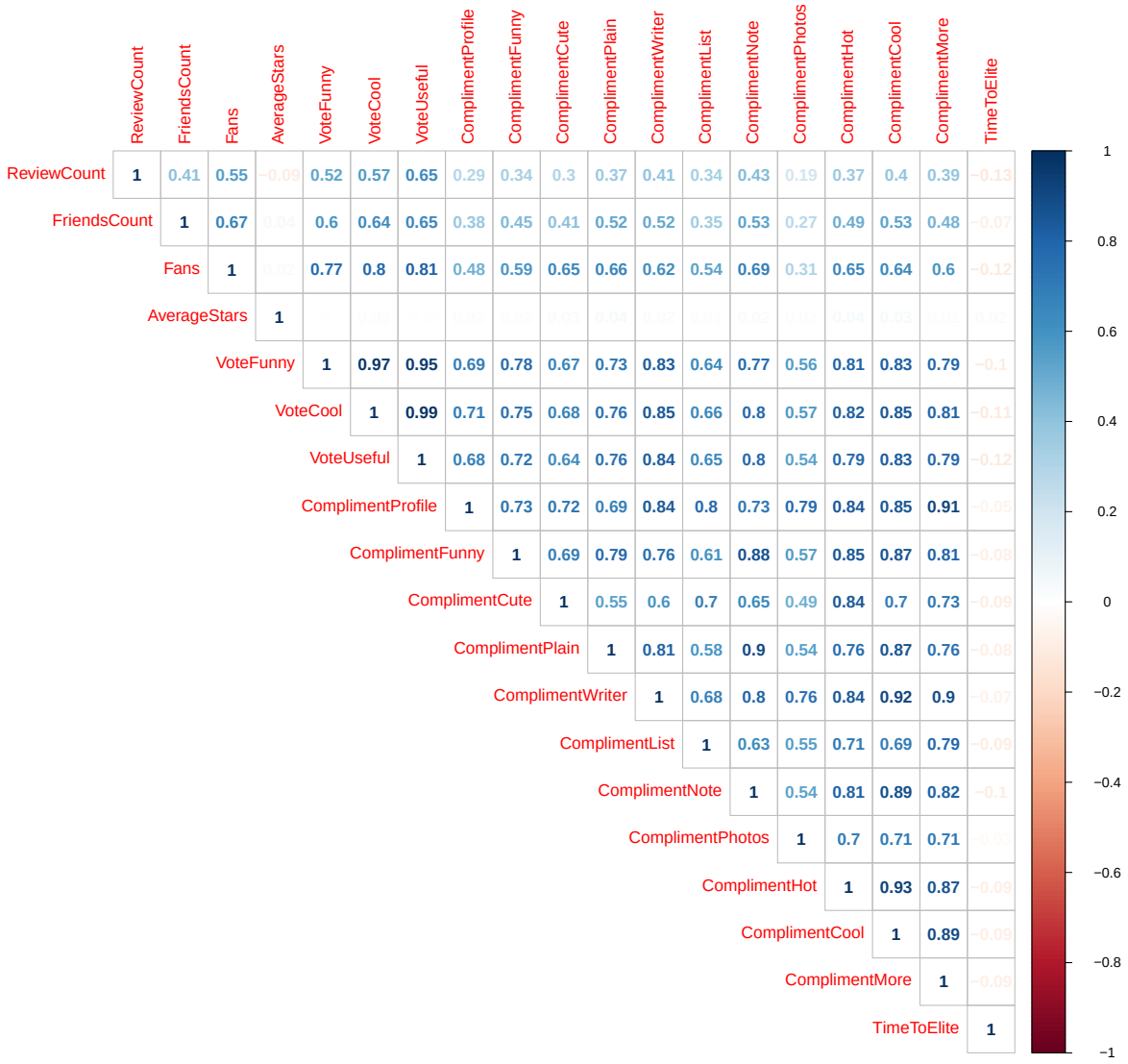
```
# Convert string to date
yelpElite$YelpingSince <- as.Date(yelpElite$YelpingSince,"%Y-%m-%d")

# Years taken to become Elite
yelpElite$TimeToElite = (yelpElite$FirstEliteYear - as.integer(format(yelpElite$YelpingSince,'%Y')))
```

Make years as factors

```
# for simplicity, drop these columns
yelpElite$YelpingSince<- NULL
yelpElite$FirstEliteYear<- NULL
yelpElite$LastEliteYear<- NULL
yelpElite$EliteYearCount <- NULL
```

```
# Make training set and validation set
# set.seed (1)
# trainVector=sample(1: nrow(yelpElite), nrow(yelpElite)/2)
# train = yelpElite[trainVector, ]
# test= yelpElite[-trainVector, ]
```

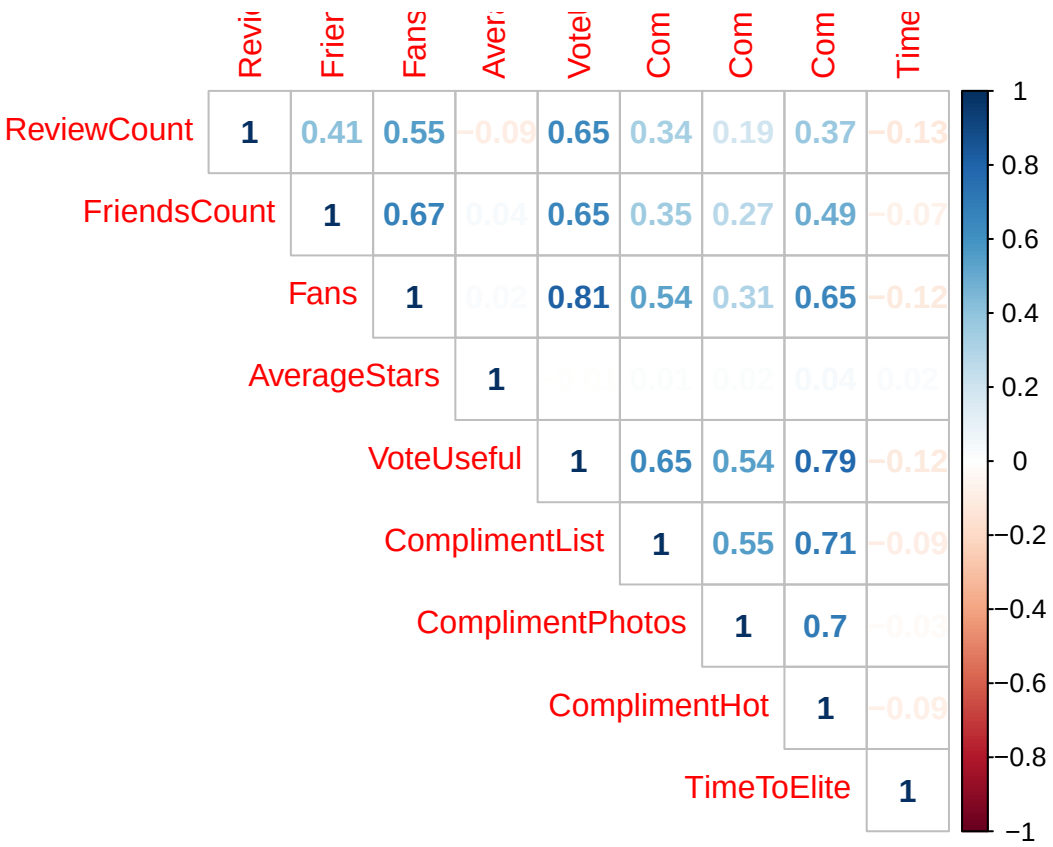
```
library('corrplot')
corrplot(cor(yelpElite), # Remove non-numeric columns
         method = "number", type = "upper")
```



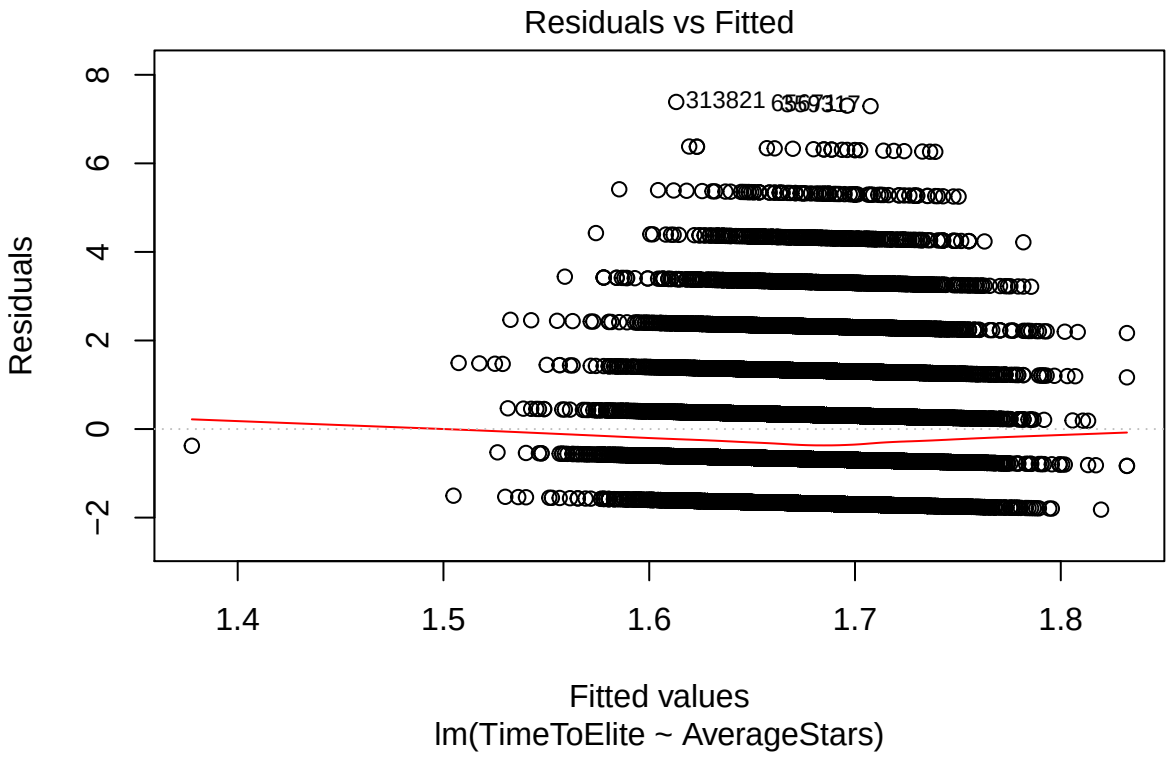
From the correlation plot, we could see that there is alot of high correlation indicating colinearlty in data.

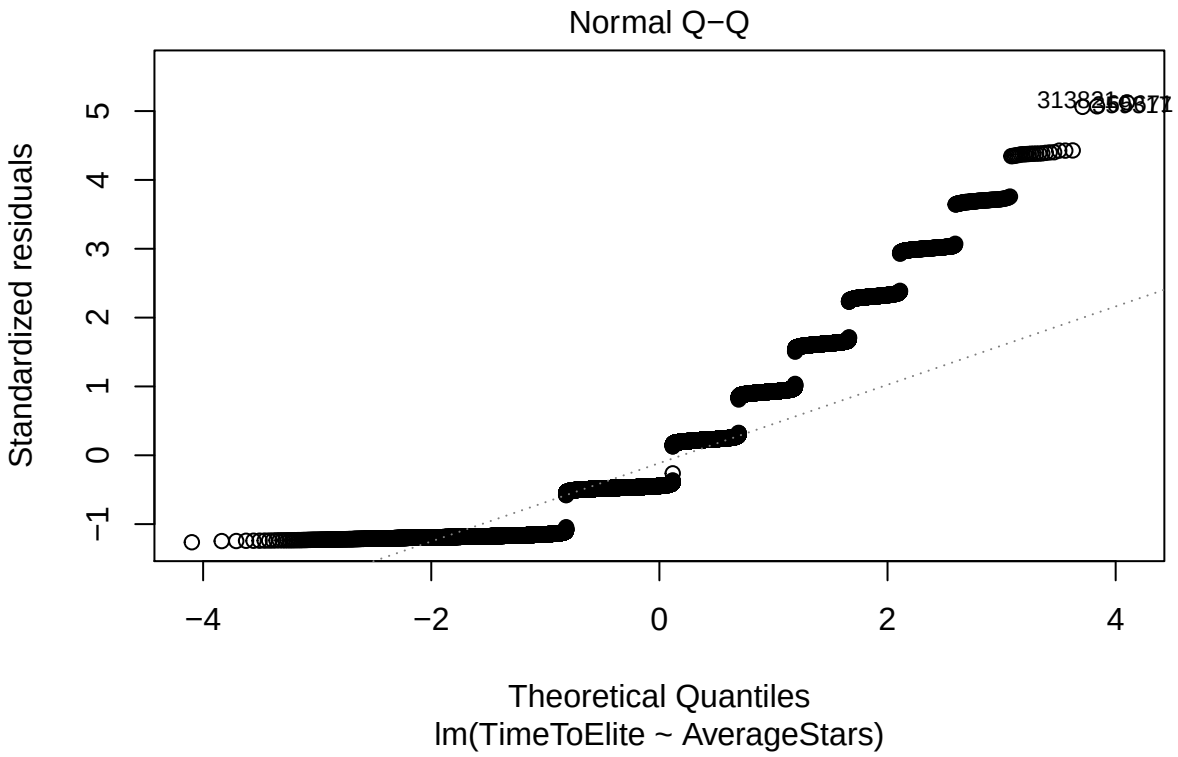
```
# correlation between votecool and voteuseful is 0.99. Hence drop VoteCool
yelpElite$VoteCool <- NULL

# correlation between voteuseful and votefunny is .96
yelpElite$VoteFunny <- NULL

#ComplimentCool is 0.93
yelpElite$ComplimentCool <-NULL

#ComplimentPlain 0.91
yelpElite$ComplimentPlain <-NULL

#ComplimentMore 0.91
yelpElite$ComplimentMore <-NULL

# All except ComplimentHot
yelpElite$ComplimentProfile <-NULL
yelpElite$ComplimentCute <-NULL
#train$ComplimentList <-NULL
yelpElite$ComplimentNote <-NULL
yelpElite$ComplimentCool <-NULL
yelpElite$ComplimentFunny <-NULL
yelpElite$ComplimentWriter <- NULL
```

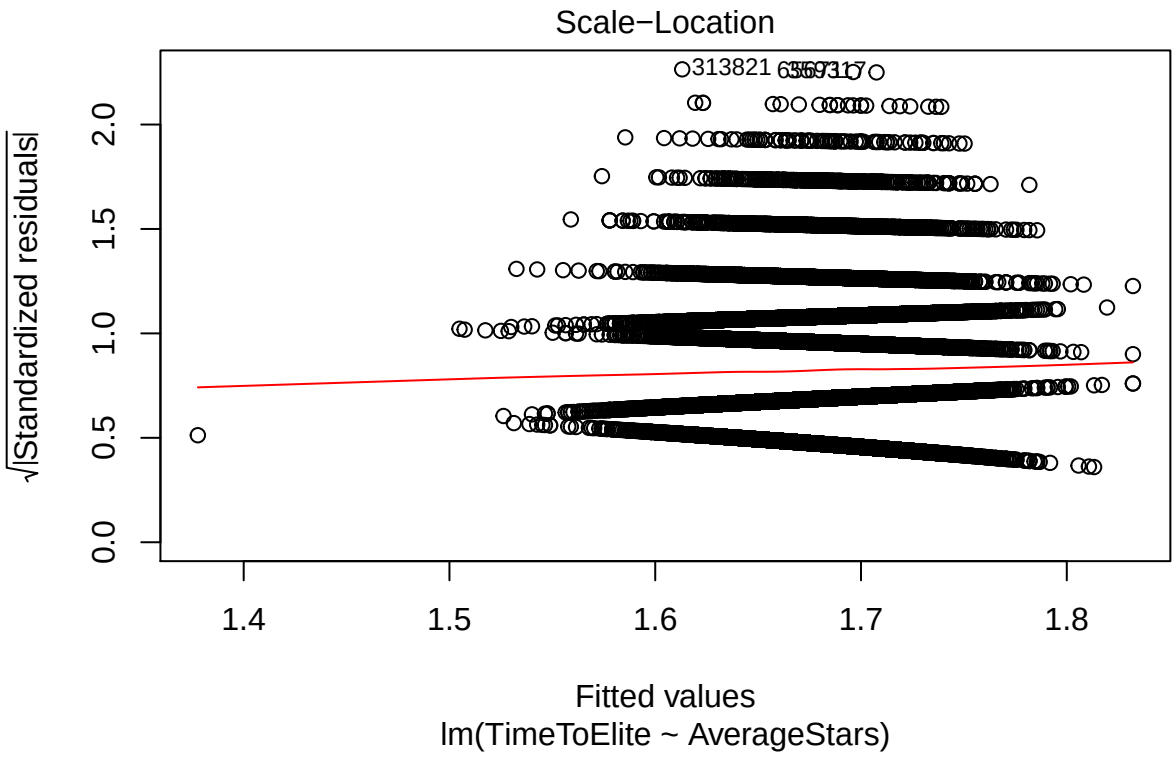
```
library('corrplot')
corrplot(cor(yelpElite), # Remove non-numeric columns
         method = "number", type = "upper")
```

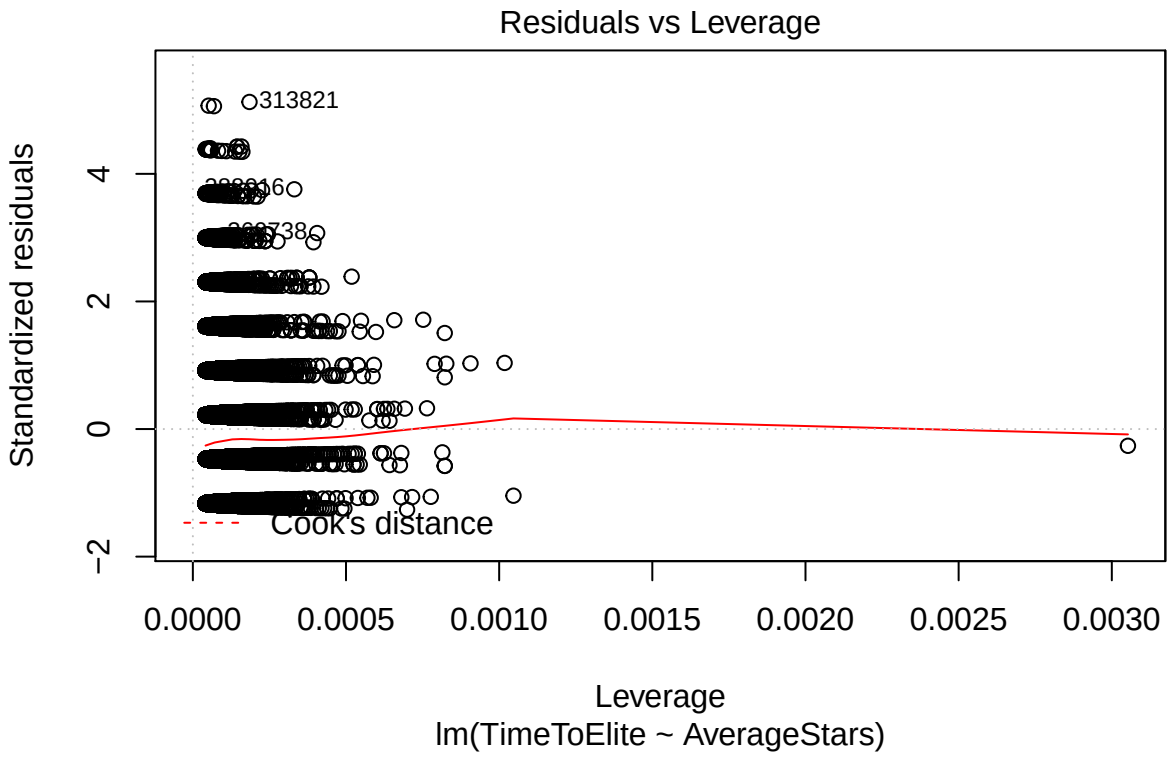



```
lm.AverageStars = lm(TimeToElite ~AverageStars, data = yelpElite)
plot(lm.AverageStars)
```

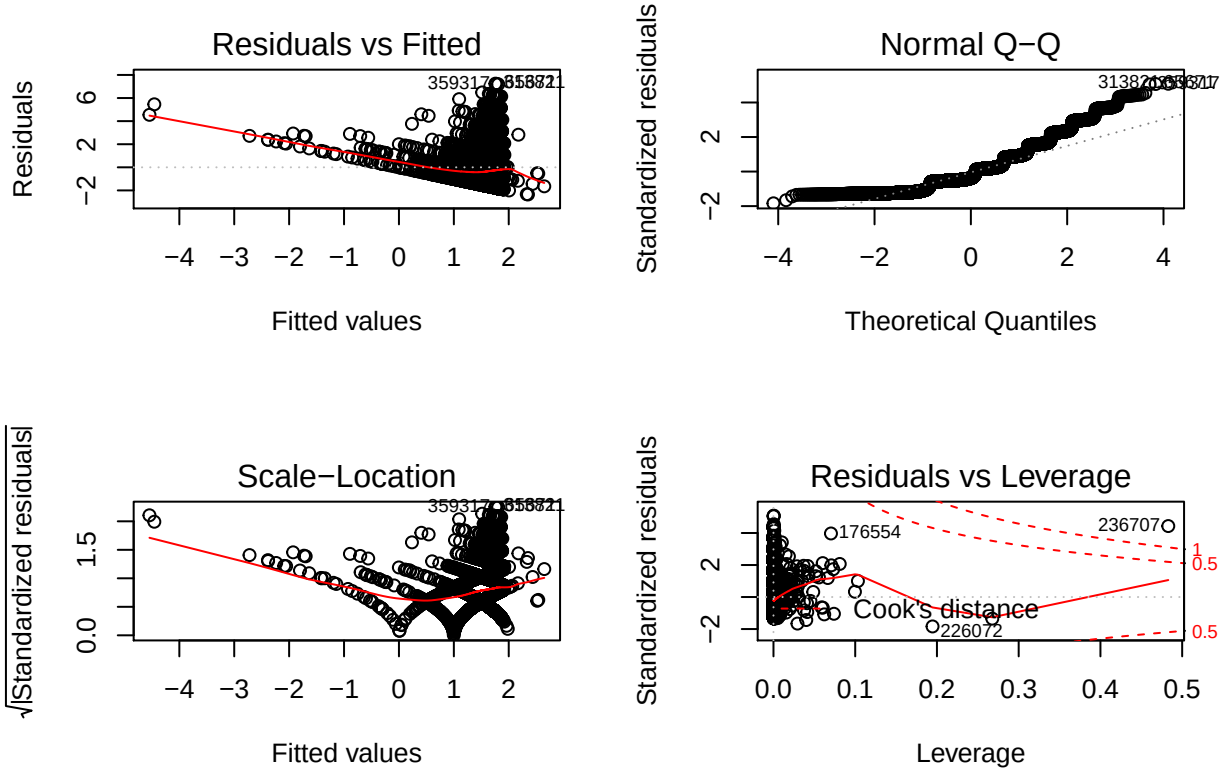

Residuals vs Fitted
Residuals
Fitted values
lm(TimeToElite ~ AverageStars)
313821
1.4
1.5
1.6
1.7
1.8
−2
0
2
4
6
8


Normal Q−Q
Standardized residuals
−1
0
1
2
3
4
5
−4
−2
0
2
4
Theoretical Quantiles
lm(TimeToElite ~ AverageStars)


Scale−Location
313821
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
1.4
1.5
1.6
1.7
1.8
Fitted values
lm(TimeToElite ~ AverageStars)




```
lm.review = lm(TimeToElite ~., data = yelpElite)
par(mfrow=c(2,2))
plot(lm.review)
```



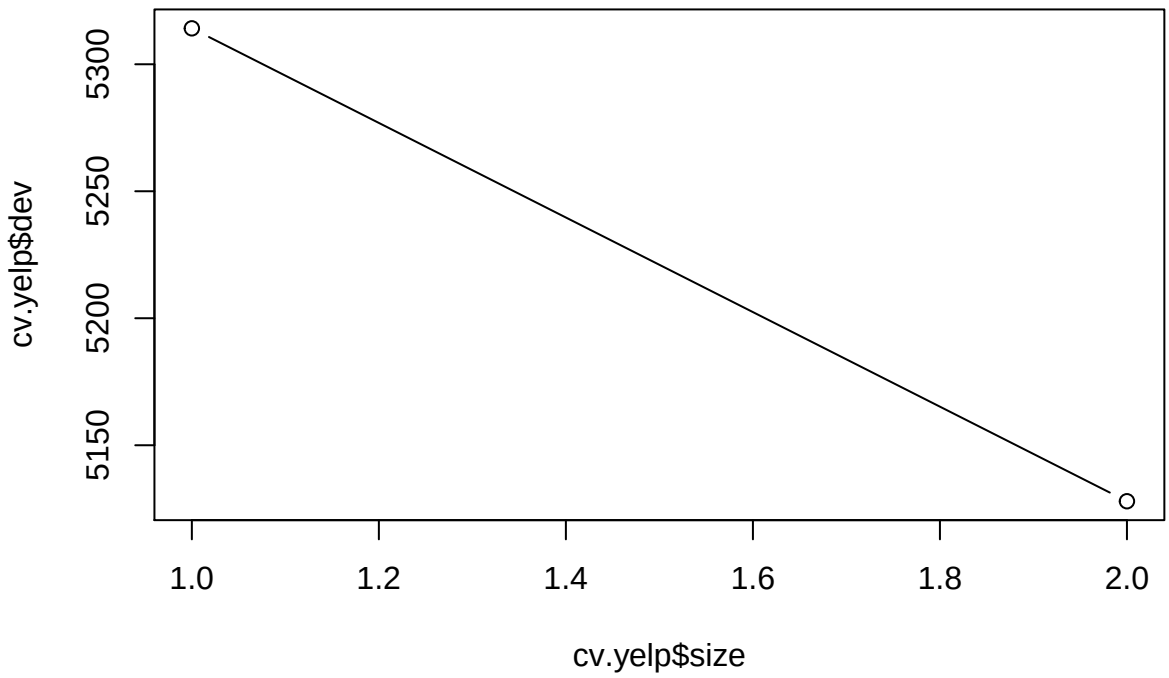

```
yelpEliteln <-yelpElite

yelpEliteln$ReviewCount <- log(yelpEliteln$ReviewCount)
yelpEliteln$FriendsCount <- log(yelpEliteln$FriendsCount)
yelpEliteln$Fans <- log(yelpEliteln$Fans)
yelpEliteln$VoteUseful <- log(yelpEliteln$VoteUseful)
yelpEliteln$ComplimentList <- log(yelpEliteln$ComplimentList)
yelpEliteln$ComplimentPhotos <- log(yelpEliteln$ComplimentPhotos)
yelpEliteln$ComplimentHot <- log(yelpEliteln$ComplimentHot)


# Remove infinite

yelpEliteln<- yelpEliteln[is.finite(yelpEliteln$ReviewCount) & is.finite(yelpEliteln$FriendsCount)
                      & is.finite(yelpEliteln$Fans)& is.finite(yelpEliteln$VoteUseful)
                      & is.finite(yelpEliteln$ComplimentList)
                      & is.finite(yelpEliteln$ComplimentHot)
                      & is.finite(yelpEliteln$ComplimentPhotos)
                    , ]


set.seed (1)
trainVector=sample(1: nrow(yelpEliteln), nrow(yelpEliteln)/2)
train = yelpEliteln[trainVector, ]
test= yelpEliteln[-trainVector, ]
```

Create the tree

```
library(tree)
tree.train = tree(TimeToElite ~ .,  data= train)
plot(tree.train)
text(tree.train ,pretty =0)
```

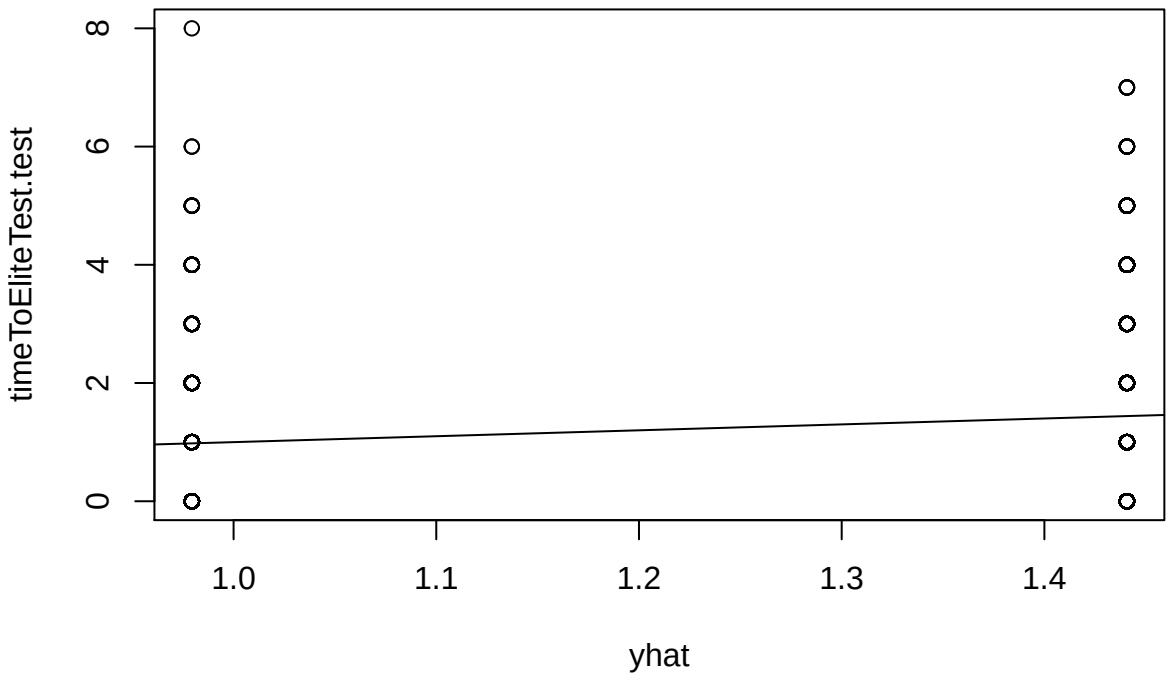
Do Cross-validation

```
cv.yelp =cv.tree(tree.train )
plot(cv.yelp$size ,cv.yelp$dev ,type='b')
```



Predict MSE

```
# PERFORMING TEST PREDICTIONS ON UNPRUNED TREE
yhat=predict(tree.train, newdata=test)
timeToEliteTest.test= test[,"TimeToElite"]
plot(yhat ,timeToEliteTest.test)
abline (0,1)
```




```
testMSE = mean((yhat-timeToEliteTest.test)^2)
testMSE
```

```
## [1] 1.38647
```

```
sqrt(testMSE)
```

```
## [1] 1.177485
```

Perform bagging

```
library(randomForest)
```

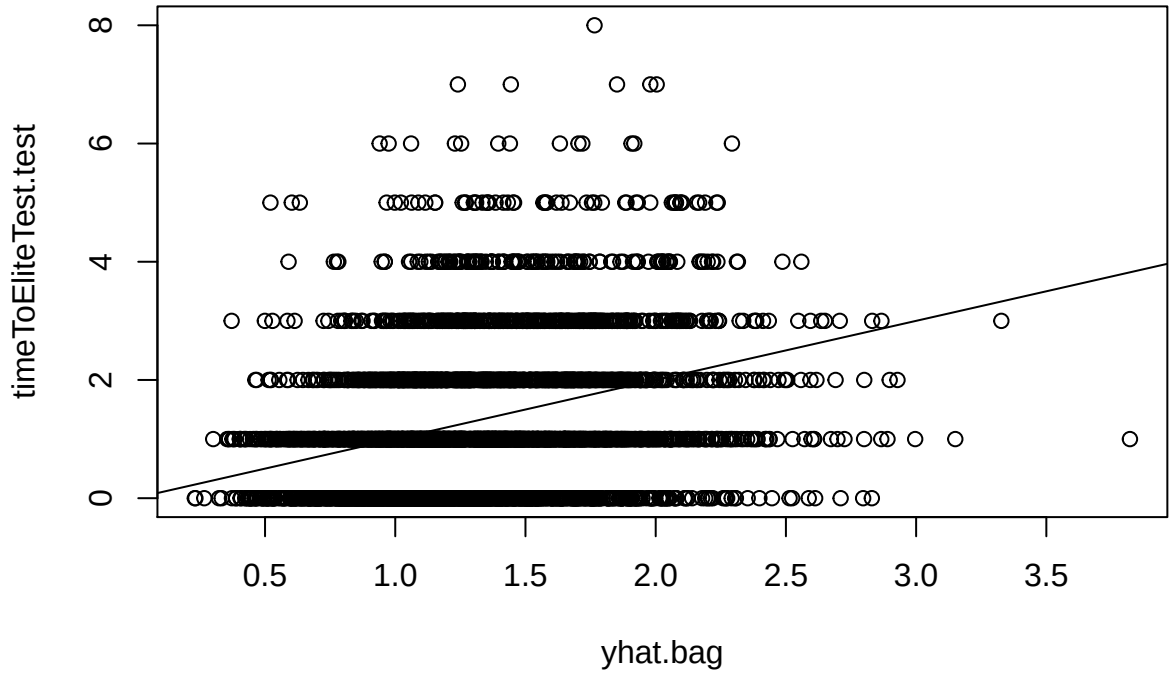
```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(1)
bag.yelp=randomForest(TimeToElite ~., data = train, mtry=8, importance =TRUE)
bag.yelp
```

```
##
## Call:
##  randomForest(formula = TimeToElite ~ ., data = train, mtry = 8,      importance = TRUE)
```

```
##                 Type of random forest: regression
##                       Number of trees: 500
## No. of variables tried at each split: 8
##
##           Mean of squared residuals: 1.440553
##                     % Var explained: 1.12
```

```
# PERFORMING TEST PREDICTIONS ON BAGGED TREE
yhat.bag=predict(bag.yelp, newdata=test)

plot(yhat.bag ,timeToEliteTest.test)
abline (0,1)
```




```
testMSE = mean((yhat.bag-timeToEliteTest.test)^2)
testMSE
```

```
## [1] 1.385713
```

```
sqrt(testMSE)
```

```
## [1] 1.177163
```

Bgging didnt improve anything

Lets do random forest

```r
# PERFORMING RANDOM FOREST
set.seed(1)
rf.yelp =randomForest(TimeToElite ~., data=train, importance =TRUE)
rf.yelp
```

```
##
## Call:
##  randomForest(formula = TimeToElite ~ ., data = train, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 1.40212
##                     % Var explained: 3.76
```

```r
# PERFORMING TEST PREDICTIONS ON RANDOM FOREST
yhat.rf = predict(rf.yelp ,newdata=test)
testMSE = mean((yhat.rf - timeToEliteTest.test)^2)
testMSE
```

```
## [1] 1.359728
```

```r
sqrt(testMSE)
```

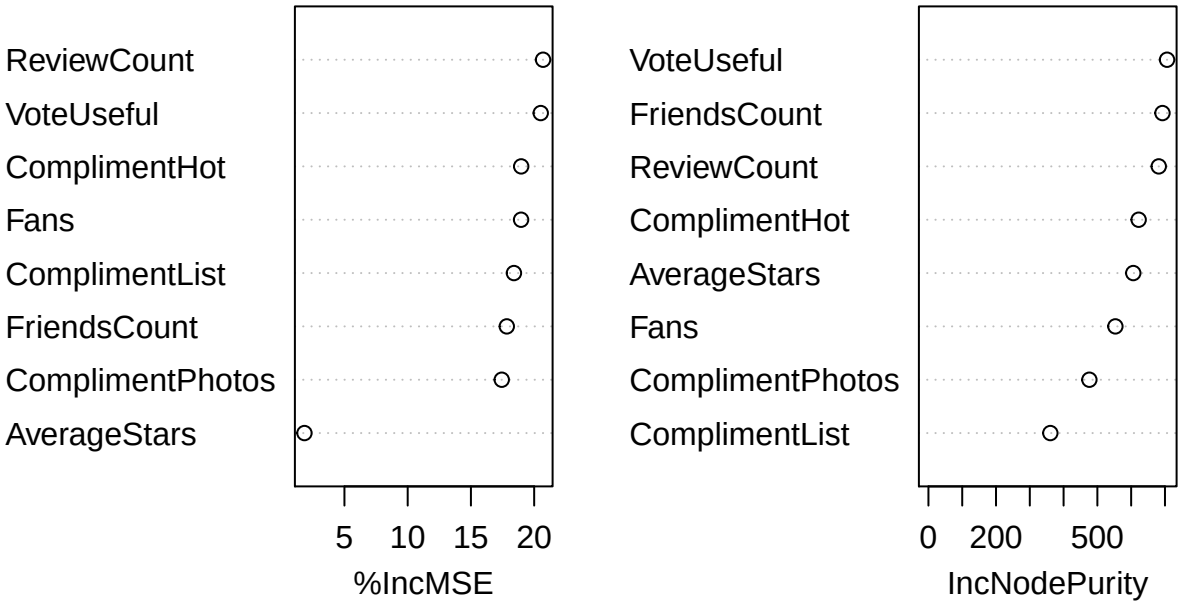
```
## [1] 1.166074
```

```r
importance(rf.yelp)
```

```
##                    %IncMSE IncNodePurity
## ReviewCount      20.702477      681.6809
## FriendsCount     17.835919      692.6185
## Fans             18.964975      553.2576
## AverageStars      1.838074      606.2120
## VoteUseful       20.518946      706.2012
## ComplimentList   18.400678      360.7446
## ComplimentPhotos 17.437022      476.3120
## ComplimentHot    18.974307      622.1477
```

```r
varImpPlot(rf.yelp)
```

rf.yelp

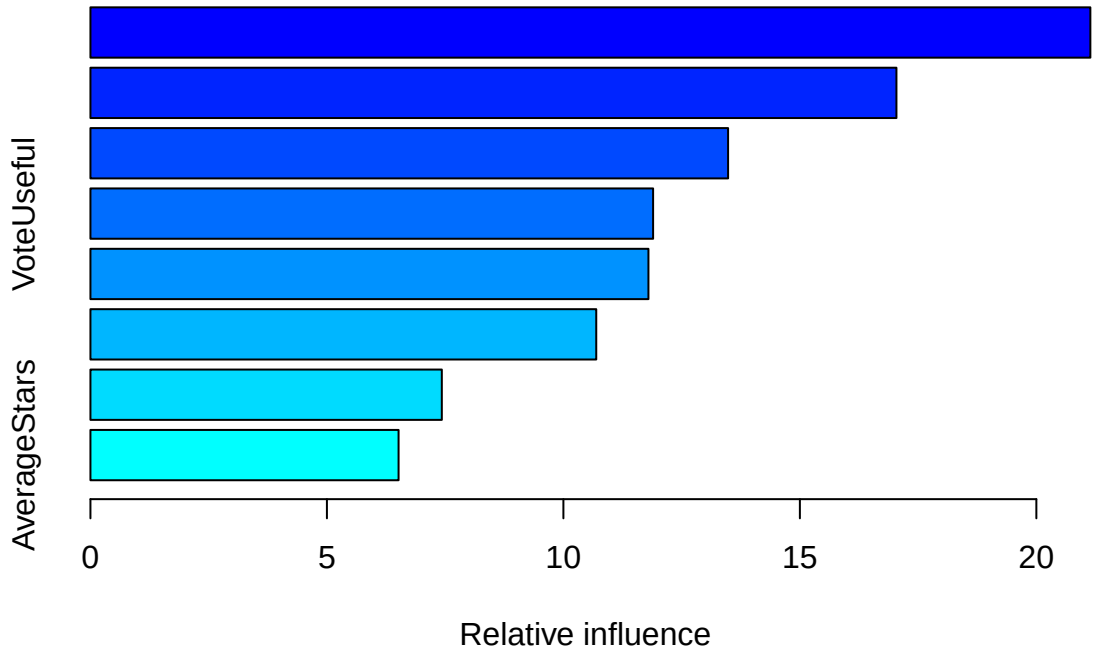


boost is the secret

```
library(gbm)
```

```
## Loading required package: survival
## Loading required package: lattice
## Loading required package: splines
## Loading required package: parallel
## Loaded gbm 2.1.1
```

```
set.seed(1)
boost.yelp=gbm(TimeToElite~.,data = train, distribution="gaussian", n.trees =5000, interaction.depth = 4
# Summary draws the relative influence plot and relative influence statistics
summary(boost.yelp)
```




```
##                               var   rel.inf
## ComplimentList     ComplimentList 21.141767
## FriendsCount         FriendsCount 17.041596
## ComplimentHot       ComplimentHot 13.482239
## VoteUseful             VoteUseful 11.896983
## ComplimentPhotos ComplimentPhotos 11.798063
## ReviewCount           ReviewCount 10.694879
## Fans                         Fans  7.429253
## AverageStars         AverageStars  6.515218
```

```
# PERFORMING TEST PREDICTIONS ON BOOSTING
yhat.boost=predict(boost.yelp, newdata=test, n.trees =5000)
testMSE = mean((yhat.boost - timeToEliteTest.test)^2)
testMSE
```

```
## [1] 1.332325
```

```
sqrt(testMSE)
```

```
## [1] 1.154264
```