Xbox game pass

R.N.Madushika

5/8/2022

```r  
set.seed(1233453)

```r  
library(tidyverse)  
library(BAS)  
library(mice)  
library(dlookr)  
library(GGally)  
library(caret)

library(readr)  
Games <- read\_csv("C:/Users/Mani/Desktop/ST406/archive (7)/Gamepass\_Games\_v1.csv")  
View(Games)  
Games <- data.frame(Games)

#Structure of the dataset  
str(Games)

## 'data.frame': 455 obs. of 9 variables:  
## $ GAME : chr "Mass Effect Legendary Edition" "The Elder Scrolls V: Skyrim Special Edition" "Mass Effect 2" "Stardew Valley" ...  
## $ RATIO : chr "1.87" "1.97" "1.34" "3.04" ...  
## $ GAMERS : num 84143 213257 221178 51530 71981 ...  
## $ COMP.. : num 4.1 8 9.6 1 15.6 6.1 9.8 3.8 0.9 1.9 ...  
## $ TIME : chr "100-120 hours" "80-100 hours" "50-60 hours" "150-200 hours" ...  
## $ RATING : num 4.8 4.7 4.7 4.7 4.7 4.6 4.6 4.6 4.6 4.6 ...  
## $ ADDED : chr "06 Jan 22" "15 Dec 20" "09 Nov 20" "02 Dec 21" ...  
## $ True\_Achievement: num 5442 3055 1819 3036 1678 ...  
## $ Game\_Score : num 2915 1550 1355 1000 1000 ...

head(Games)

## GAME RATIO GAMERS COMP.. TIME  
## 1 Mass Effect Legendary Edition 1.87 84143 4.1 100-120 hours  
## 2 The Elder Scrolls V: Skyrim Special Edition 1.97 213257 8.0 80-100 hours  
## 3 Mass Effect 2 1.34 221178 9.6 50-60 hours  
## 4 Stardew Valley 3.04 51530 1.0 150-200 hours  
## 5 It Takes Two 1.68 71981 15.6 12-15 hours  
## 6 Hades 2.40 83710 6.1 60-80 hours  
## RATING ADDED True\_Achievement Game\_Score  
## 1 4.8 06 Jan 22 5442 2915  
## 2 4.7 15 Dec 20 3055 1550  
## 3 4.7 09 Nov 20 1819 1355  
## 4 4.7 02 Dec 21 3036 1000  
## 5 4.7 03 Nov 21 1678 1000  
## 6 4.6 18 Jun 21 2404 1000

summary(Games)

## GAME RATIO GAMERS COMP..   
## Length:455 Length:455 Min. : 0 Min. : 0.000   
## Class :character Class :character 1st Qu.: 17781 1st Qu.: 0.600   
## Mode :character Mode :character Median : 52268 Median : 1.900   
## Mean : 83540 Mean : 6.345   
## 3rd Qu.:116796 3rd Qu.: 6.100   
## Max. :455839 Max. :84.700   
##   
## TIME RATING ADDED True\_Achievement  
## Length:455 Min. :2.000 Length:455 Min. : 252   
## Class :character 1st Qu.:3.400 Class :character 1st Qu.: 2102   
## Mode :character Median :3.700 Mode :character Median : 3231   
## Mean :3.702 Mean : 4693   
## 3rd Qu.:4.100 3rd Qu.: 4995   
## Max. :4.800 Max. :37178   
## NA's :3   
## Game\_Score   
## Min. : 200   
## 1st Qu.:1000   
## Median :1000   
## Mean :1209   
## 3rd Qu.:1212   
## Max. :7000   
##

#data cleaning  
  
#Remove duplicates  
Games <- Games[!duplicated(Games$GAME), ]  
  
#Check for missing values  
sum(is.na(Games))

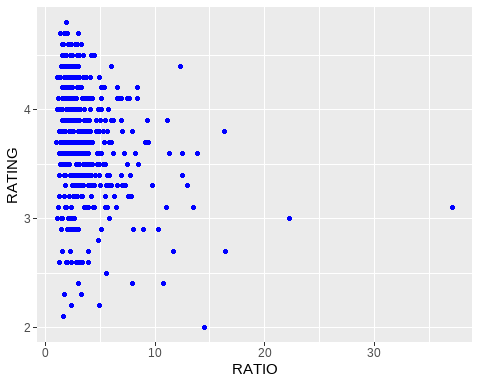
## [1] 38

#remove the column with high missing values  
Games = subset(Games, select = -c(TIME) )  
#Games

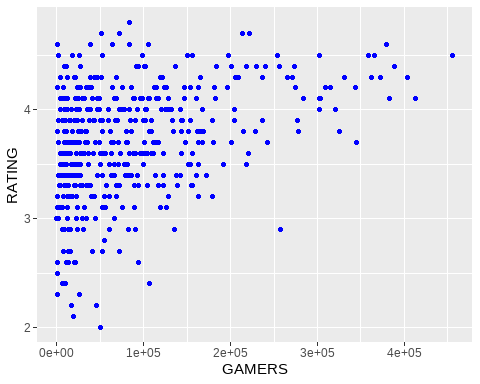
#remove rows with missing values  
Games <- Games %>% drop\_na()  
#Games

Games$RATIO <- as.numeric(Games$RATIO)  
#Games

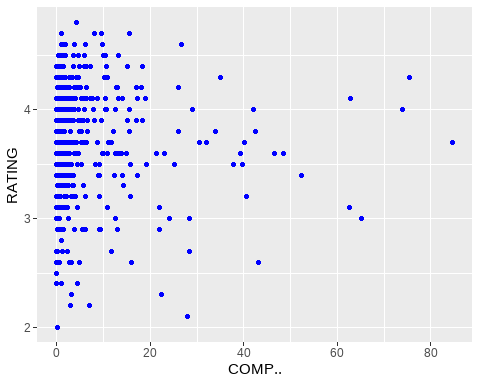
#Exploratory data analysis  
describe(Games)  
#Scatter plots  
ggplot(Games, aes(x=RATIO, y=RATING)) + geom\_point(color="blue")



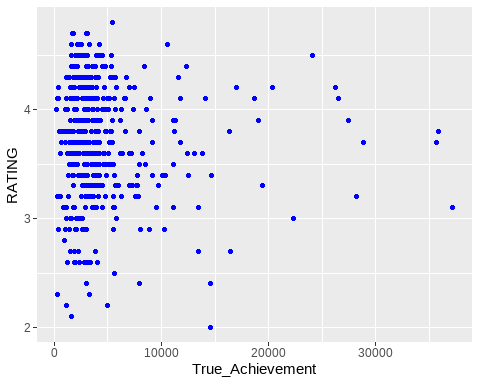
ggplot(Games, aes(x=GAMERS, y=RATING)) + geom\_point(color="blue")



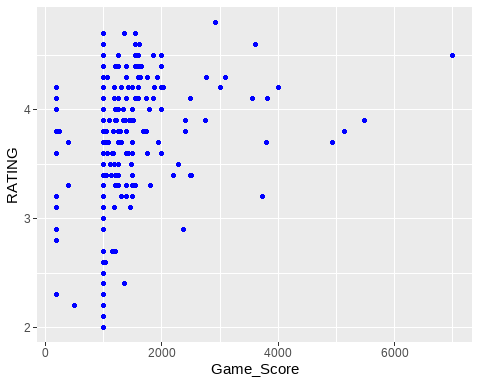
ggplot(Games, aes(x=COMP.., y=RATING)) + geom\_point(color="blue")



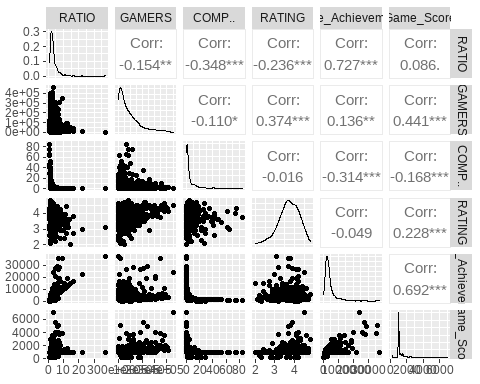
ggplot(Games, aes(x=True\_Achievement,y=RATING)) + geom\_point(color="blue")



ggplot(Games, aes(x=Game\_Score, y=RATING)) + geom\_point(color="blue")



#Find all correlations  
ggpairs(Games[-c(1,6)],)



#Building a model  
#split data into training and test data sets  
indxTrain <- createDataPartition(y = Games$RATING,p = 0.75,list = FALSE)  
training <- Games[indxTrain,]  
testing <- Games[-indxTrain,]  
#Check dimensions of the split  
prop.table(table(Games$RATING)) \* 100

##   
## 2 2.1 2.2 2.3 2.4 2.5 2.6 2.7   
## 0.2227171 0.2227171 0.4454343 0.4454343 0.6681514 0.2227171 1.7817372 1.3363029   
## 2.8 2.9 3 3.1 3.2 3.3 3.4 3.5   
## 0.2227171 2.6726058 1.7817372 3.5634744 3.7861915 5.5679287 6.0133630 5.5679287   
## 3.6 3.7 3.8 3.9 4 4.1 4.2 4.3   
## 9.7995546 7.7951002 6.4587973 7.1269488 6.4587973 8.0178174 5.3452116 4.8997773   
## 4.4 4.5 4.6 4.7 4.8   
## 3.7861915 3.1180401 1.5590200 0.8908686 0.2227171

prop.table(table(training$RATING)) \* 100

##   
## 2.1 2.2 2.3 2.4 2.6 2.7 2.8 2.9   
## 0.295858 0.591716 0.295858 0.591716 1.479290 1.183432 0.295858 2.366864   
## 3 3.1 3.2 3.3 3.4 3.5 3.6 3.7   
## 2.366864 4.142012 4.142012 5.325444 5.917160 6.213018 9.763314 7.100592   
## 3.8 3.9 4 4.1 4.2 4.3 4.4 4.5   
## 6.213018 7.988166 6.804734 7.100592 5.325444 4.733728 3.550296 2.958580   
## 4.6 4.7 4.8   
## 1.775148 1.183432 0.295858

prop.table(table(testing$RATING)) \* 100

##   
## 2 2.3 2.4 2.5 2.6 2.7 2.9   
## 0.9009009 0.9009009 0.9009009 0.9009009 2.7027027 1.8018018 3.6036036   
## 3.1 3.2 3.3 3.4 3.5 3.6 3.7   
## 1.8018018 2.7027027 6.3063063 6.3063063 3.6036036 9.9099099 9.9099099   
## 3.8 3.9 4 4.1 4.2 4.3 4.4   
## 7.2072072 4.5045045 5.4054054 10.8108108 5.4054054 5.4054054 4.5045045   
## 4.5 4.6   
## 3.6036036 0.9009009

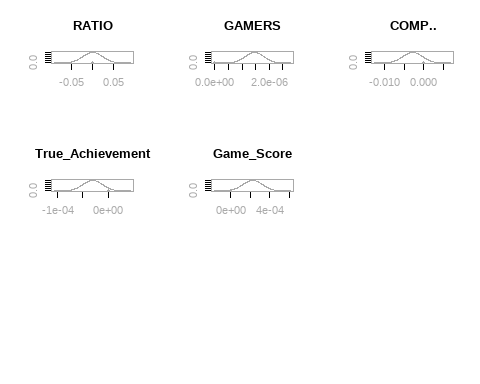
#Bayesian Multiple Regression Model  
  
Games.bas = bas.lm(RATING ~ ., data = training[-c(1,6)], prior = "BIC",  
  
modelprior =tr.beta.binomial(1,1,5),  
include.always = ~ .,  
n.models = 1)  
  
Games.bas

##   
## Call:  
## bas.lm(formula = RATING ~ ., data = training[-c(1, 6)], n.models = 1,   
## prior = "BIC", modelprior = tr.beta.binomial(1, 1, 5), include.always = ~.)  
##   
##   
## Marginal Posterior Inclusion Probabilities:   
## Intercept RATIO GAMERS COMP..   
## 1 1 1 1   
## True\_Achievement Game\_Score   
## 1 1

Games.coef = coef(Games.bas)  
Games.coef

##   
## Marginal Posterior Summaries of Coefficients:   
##   
## Using BMA   
##   
## Based on the top 1 models   
## post mean post SD post p(B != 0)  
## Intercept 3.707e+00 2.528e-02 1.000e+00   
## RATIO 1.232e-04 2.276e-02 1.000e+00   
## GAMERS 1.460e-06 3.392e-07 1.000e+00   
## COMP.. -2.808e-03 2.477e-03 1.000e+00   
## True\_Achievement -3.064e-05 1.914e-05 1.000e+00   
## Game\_Score 2.255e-04 9.987e-05 1.000e+00

par(mfrow = c(3, 3), col.lab = "darkgrey", col.axis = "darkgrey", col = "darkgrey")  
plot(Games.coef, subset = 2:6, ask = F)



#Credible interval  
confint(Games.coef,parm=2:6)

## 2.5% 97.5% beta  
## RATIO -4.465320e-02 4.489955e-02 1.231774e-04  
## GAMERS 7.924027e-07 2.127023e-06 1.459713e-06  
## COMP.. -7.680424e-03 2.064663e-03 -2.807880e-03  
## True\_Achievement -6.829666e-05 7.018107e-06 -3.063928e-05  
## Game\_Score 2.901776e-05 4.219359e-04 2.254768e-04  
## attr(,"Probability")  
## [1] 0.95  
## attr(,"class")  
## [1] "confint.bas"

##Since 0 is included within credible interval ratio,comp.. and true achievement are not significant to the model  
  
  
out = confint(Games.coef)[,1:2]  
# Extract the upper and lower bounds of the credible intervals  
names = c("posterior mean", "posterior std", colnames(out))  
out = cbind(Games.coef$postmean, Games.coef$postsd, out)  
colnames(out) = names  
round(out, 2)

## posterior mean posterior std 2.5% 97.5%  
## Intercept 3.71 0.03 3.66 3.76  
## RATIO 0.00 0.02 -0.04 0.04  
## GAMERS 0.00 0.00 0.00 0.00  
## COMP.. 0.00 0.00 -0.01 0.00  
## True\_Achievement 0.00 0.00 0.00 0.00  
## Game\_Score 0.00 0.00 0.00 0.00

#select the best model  
#AIc  
n=nrow(training)  
Games.lm=lm(RATING ~.,data=training[-c(1,6)])  
Games.step=step(Games.lm,k=log(n))

## Start: AIC=-489.12  
## RATING ~ RATIO + GAMERS + COMP.. + True\_Achievement + Game\_Score  
##   
## Df Sum of Sq RSS AIC  
## - RATIO 1 0.0000 71.706 -494.95  
## - COMP.. 1 0.2775 71.983 -493.64  
## - True\_Achievement 1 0.5533 72.259 -492.35  
## - Game\_Score 1 1.1009 72.807 -489.80  
## <none> 71.706 -489.12  
## - GAMERS 1 3.9991 75.705 -476.60  
##   
## Step: AIC=-494.95  
## RATING ~ GAMERS + COMP.. + True\_Achievement + Game\_Score  
##   
## Df Sum of Sq RSS AIC  
## - COMP.. 1 0.2891 71.995 -499.41  
## <none> 71.706 -494.95  
## - Game\_Score 1 3.1389 74.845 -486.29  
## - True\_Achievement 1 3.8194 75.525 -483.23  
## - GAMERS 1 4.0001 75.706 -482.42  
##   
## Step: AIC=-499.41  
## RATING ~ GAMERS + True\_Achievement + Game\_Score  
##   
## Df Sum of Sq RSS AIC  
## <none> 71.995 -499.41  
## - Game\_Score 1 2.9720 74.967 -491.56  
## - True\_Achievement 1 3.5321 75.527 -489.04  
## - GAMERS 1 4.3084 76.303 -485.59

Games.BIC=bas.lm(RATING~.,data=training[-c(1,6)],prior="BIC",modelprior = tr.beta.binomial(1,1,5))  
Games.BIC

##   
## Call:  
## bas.lm(formula = RATING ~ ., data = training[-c(1, 6)], prior = "BIC",   
## modelprior = tr.beta.binomial(1, 1, 5))  
##   
##   
## Marginal Posterior Inclusion Probabilities:   
## Intercept RATIO GAMERS COMP..   
## 1.0000 0.4967 0.9995 0.1690   
## True\_Achievement Game\_Score   
## 0.5605 0.6407

best=which.max(Games.BIC$logmarg)  
  
bestmodel=Games.BIC$which[[best]]  
bestmodel

## [1] 0 2 4 5

bestgamma=rep(0, Games.BIC$n.vars)  
bestgamma[bestmodel+1]=1  
bestgamma

## [1] 1 0 1 0 1 1

#summary of best 5 models  
Games\_bas=bas.lm(RATING~GAMERS+True\_Achievement+Game\_Score,data=training[-c(1,6),],prior="BIC",modelprior = tr.beta.binomial(1,1,5))  
round(summary(Games\_bas),3)

## P(B != 0 | Y) model 1 model 2 model 3 model 4 model 5  
## Intercept 1.000 1.000 1.000 1.000 1.000 1.00  
## GAMERS 1.000 1.000 1.000 1.000 1.000 0.00  
## True\_Achievement 0.943 1.000 0.000 1.000 0.000 1.00  
## Game\_Score 0.924 1.000 0.000 0.000 1.000 1.00  
## BF NA 1.000 0.171 0.077 0.016 0.00  
## PostProbs NA 0.919 0.052 0.023 0.005 0.00  
## R2 NA 0.164 0.125 0.136 0.128 0.11  
## dim NA 4.000 2.000 3.000 3.000 3.00  
## logmarg NA -726.943 -728.707 -729.513 -731.109 -734.58

##BIC is lower the better. logmarg=(-1/2BIC).logmarg higher the better  
print(Games\_bas)

##   
## Call:  
## bas.lm(formula = RATING ~ GAMERS + True\_Achievement + Game\_Score,   
## data = training[-c(1, 6), ], prior = "BIC", modelprior = tr.beta.binomial(1,   
## 1, 5))  
##   
##   
## Marginal Posterior Inclusion Probabilities:   
## Intercept GAMERS True\_Achievement Game\_Score   
## 1.0000 0.9999 0.9428 0.9241

#model validation  
#Predict testing set  
Predict <- predict(Games\_bas,newdata = testing[-c(1,6)] )  
#Compute errors  
Error=testing$RATING-Predict$fit  
#Error  
#RMSE  
RMSE=sqrt(mean(Error^2))  
RMSE

## [1] 0.4662949

Games.coef2=coef(Games\_bas)  
confint(Games.coef2)

## 2.5% 97.5% beta  
## Intercept 3.654946e+00 3.753874e+00 3.701190e+00  
## GAMERS 9.326980e-07 2.340336e-06 1.598260e-06  
## True\_Achievement -3.734677e-05 0.000000e+00 -2.440489e-05  
## Game\_Score 0.000000e+00 2.955501e-04 1.819028e-04  
## attr(,"Probability")  
## [1] 0.95  
## attr(,"class")  
## [1] "confint.bas"

par(mfrow = c(2, 2), col.lab = "darkgrey", col.axis = "darkgrey", col = "darkgrey")  
plot(Games.coef2, subset = 2:4, ask = F)

