Assignment-2

Manuel Bottino

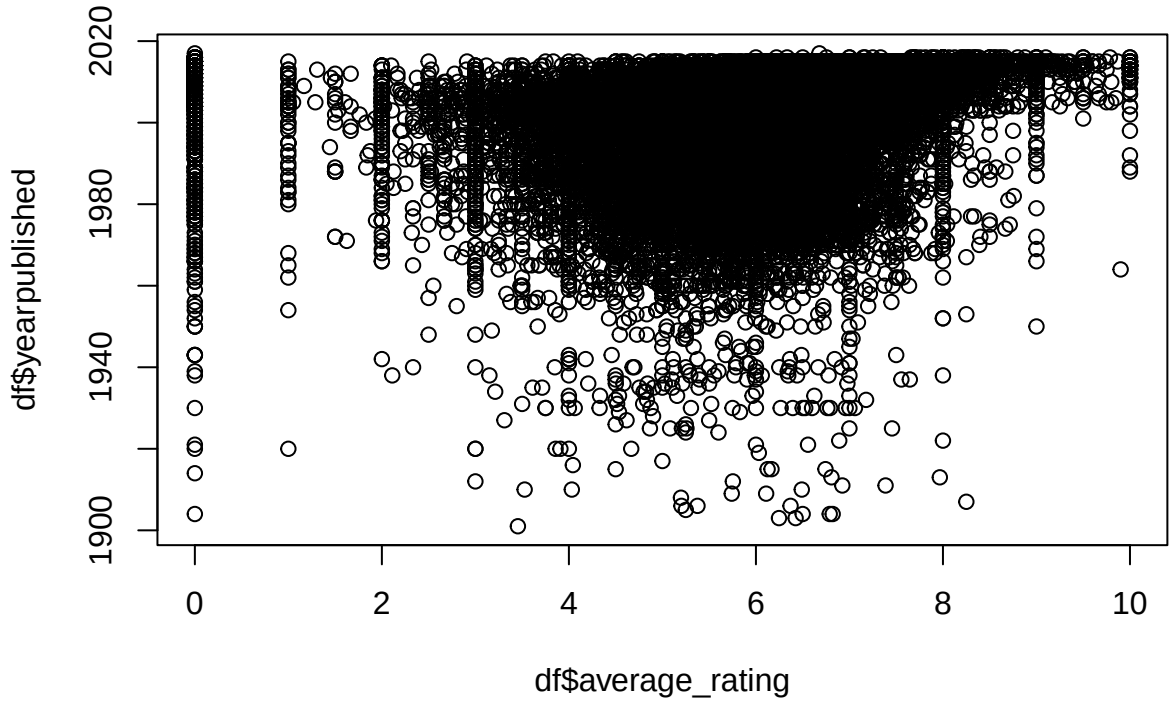
2023-05-28

Contents

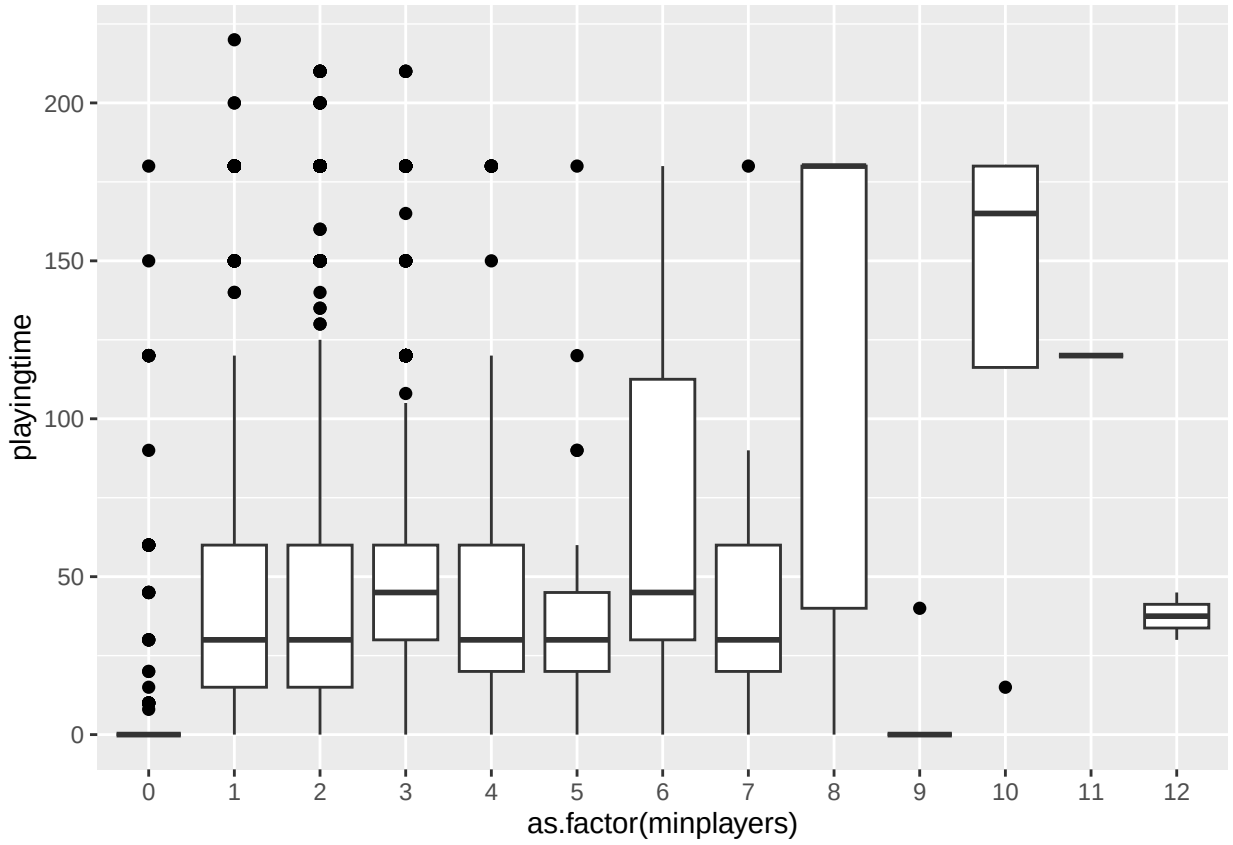
1 Excercise 1 - Risiko! is back, with friends

1.1 Point a)

```
df <- readRDS("~/Documents/GitHub/Computational-stat/Assignment 2/games_preprocessed.RDS")

plot(df$average_rating, df$yearpublished) # to see whether old board_games tend to be preferred more be
```




```
ggplot(data=df, aes(y=playingtime, x=as.factor(minplayers)))+ # As more player play are needed to start
  geom_boxplot(outlier.colour="black", outlier.shape=16,
               outlier.size=2, notch=FALSE)
```



Point b)

```
df= df[,-1:-2]
df=data.frame(scale(df))
```

```
sample_cov=cov(df)
cor(df)
```

```
##                yearpublished  minplayers  maxplayers playingtime     minage
## yearpublished     1.00000000  0.03585626  0.03112545 -0.06572597 0.09314918
## minplayers        0.03585626  1.00000000  0.19166864  0.04429612 0.09851689
## maxplayers        0.03112545  0.19166864  1.00000000 -0.01204945 0.05626623
## playingtime      -0.06572597  0.04429612 -0.01204945  1.00000000 0.28407403
## minage            0.09314918  0.09851689  0.05626623  0.28407403 1.00000000
## average_rating    0.21657330 -0.01705543 -0.02509443  0.18694736 0.16096705
## total_owners      0.04441719  0.02724071  0.01574204  0.13281225 0.15929911
## average_weight    0.04899808 -0.06353175 -0.09918355  0.43645196 0.21685937
##                average_rating total_owners average_weight
## yearpublished      0.21657330   0.04441719     0.04899808
## minplayers        -0.01705543   0.02724071    -0.06353175
## maxplayers        -0.02509443   0.01574204    -0.09918355
## playingtime        0.18694736   0.13281225     0.43645196
## minage             0.16096705   0.15929911     0.21685937
## average_rating     1.00000000   0.16819425     0.33090040
## total_owners       0.16819425   1.00000000     0.13106392
## average_weight     0.33090040   0.13106392     1.00000000
```

```
# We can get the same result using cov2cor
cov2cor(sample_cov)
```

```
##                yearpublished  minplayers  maxplayers playingtime     minage
## yearpublished     1.00000000  0.03585626  0.03112545 -0.06572597 0.09314918
## minplayers        0.03585626  1.00000000  0.19166864  0.04429612 0.09851689
## maxplayers        0.03112545  0.19166864  1.00000000 -0.01204945 0.05626623
## playingtime      -0.06572597  0.04429612 -0.01204945  1.00000000 0.28407403
## minage            0.09314918  0.09851689  0.05626623  0.28407403 1.00000000
## average_rating    0.21657330 -0.01705543 -0.02509443  0.18694736 0.16096705
## total_owners      0.04441719  0.02724071  0.01574204  0.13281225 0.15929911
## average_weight    0.04899808 -0.06353175 -0.09918355  0.43645196 0.21685937
##                average_rating total_owners average_weight
## yearpublished      0.21657330   0.04441719     0.04899808
## minplayers        -0.01705543   0.02724071    -0.06353175
## maxplayers        -0.02509443   0.01574204    -0.09918355
## playingtime        0.18694736   0.13281225     0.43645196
## minage             0.16096705   0.15929911     0.21685937
## average_rating     1.00000000   0.16819425     0.33090040
## total_owners       0.16819425   1.00000000     0.13106392
## average_weight     0.33090040   0.13106392     1.00000000
```

Since we have scaled all the columns with variance equal to 1, it implies that the covariance-variance matrix is equal to the correlation matrix. Fonte: https://math.stackexchange.com/questions/3780344/under-what-conditions-will-the-covariance-matrix-be-identical-to-the-correlation

1.2 Point c)

```
B=1000

cov_var_sample=array(NA, dim=c(ncol(df),ncol(df),B))

for(i in 1:B){
  cov_var_sample[,,i]=cov(slice_sample(df,n=nrow(df), replace=TRUE))
}

theta_hat_bar=apply(cov_var_sample, c(1, 2), mean)
se_cov_var=apply(cov_var_sample, c(1, 2), sd)
```

Confidence intervals 1. A first method consist on using the fact that, as ths sample size n grows large $\frac{\hat{\theta}-\theta}{SE_B(\hat{\theta})} \approx N(0,1)$

```
lower_bound=sample_cov+qnorm(c(0.025), mean=0, sd=1)*se_cov_var
upper_bound=sample_cov+qnorm(c(0.975), mean=0, sd=1)*se_cov_var
```

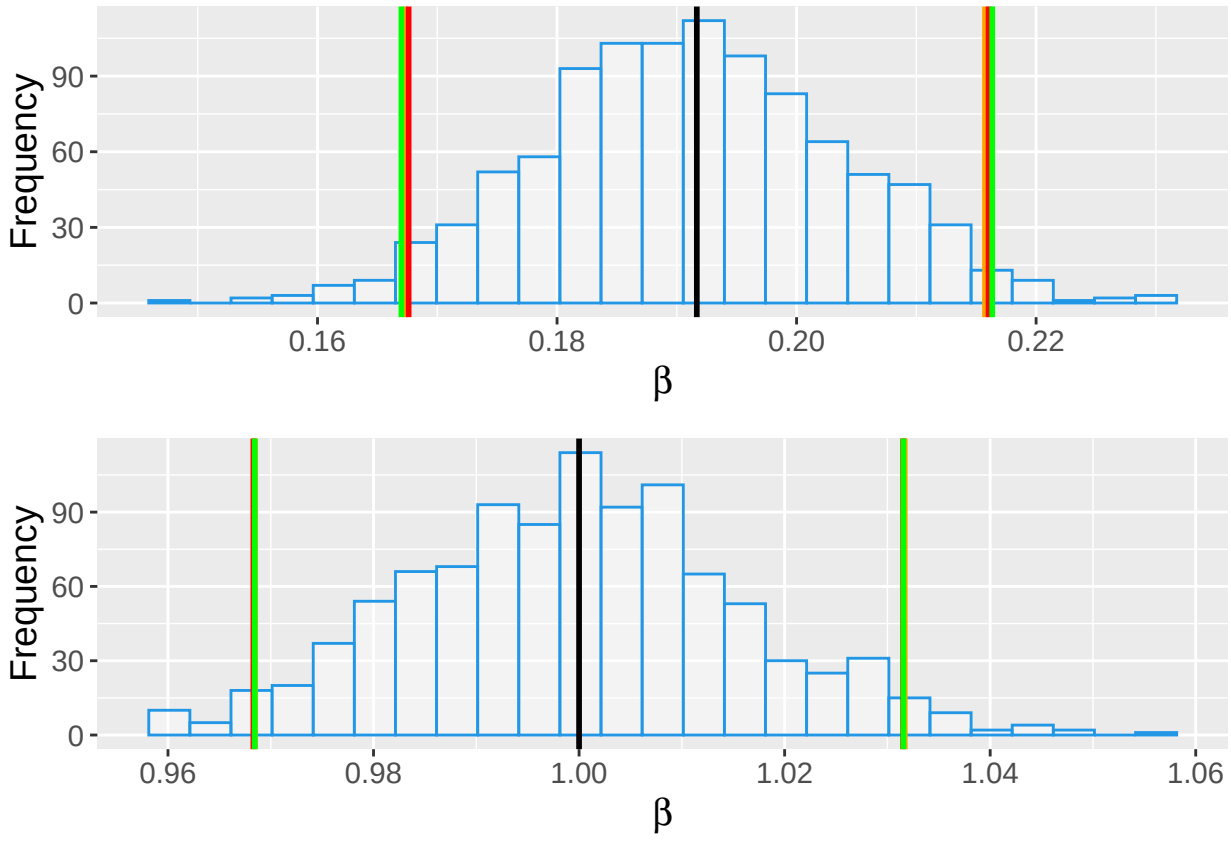
2. Use the quantile of the bootstrap sampling distribution, this method is considered more robust than the one before

```
boot_sample_variance <- data.frame(matrix(nrow = 6, ncol = 6))
colnames(boot_sample_variance) <- c("maxplayers", "playingtime", "minage", "average_rating", "total_owne

for(i in 1:6){
  for(j in 1:6){
    boot_sample_variance[i,j]=list(list(cov_var_sample[i,j,]))
  }
}
boot_sample_variance_ci <- data.frame(matrix(nrow = 6, ncol = 6))
colnames(boot_sample_variance_ci) <- c("maxplayers", "playingtime", "minage", "average_rating", "total_

for(i in 1:6){
  for(j in 1:6){
    boot_sample_variance_ci[i,j]=list(list(quantile(boot_sample_variance[i,j][[1]], probs=c(0.025,0.975)
  }
}
```

3. Using as assumption the fact that the behavior of the bias $\theta - \hat{\theta}$ is the same as $\hat{\theta} - \hat{\theta}^*$, we can compute the confidence interval of the bias-corrected estimate

```
boot_sample_variance_ci_bias <- data.frame(matrix(nrow = 6, ncol = 6))
colnames(boot_sample_variance_ci) <- c("maxplayers", "playingtime", "minage", "average_rating", "total_

for(i in 1:6){
  for(j in 1:6){
    boot_sample_variance_ci_bias[i,j]=list(list(2*sample_cov[i,j]-quantile(boot_sample_variance[i,j][[1
  }
}
```

```
plot_boot_sample_ci = function(row, column){
  ggplot()+
  geom_histogram(aes(x=boot_sample_variance[row,column][[1]]),alpha=.4,col=4,fill="white",bins=25)+
  geom_vline(aes(xintercept = sample_cov[row,column]) ,col=1, lwd = 1)+
  geom_vline(aes(xintercept = boot_sample_variance_ci[row, column][[1]][1]) ,col="orange", lwd = 1)+
  geom_vline(aes(xintercept = boot_sample_variance_ci[row, column][[1]][2]) ,col="orange", lwd = 1)+
  geom_vline(aes(xintercept = boot_sample_variance_ci_bias[row, column][[1]][1]) ,col="red", lwd = 1)+
  geom_vline(aes(xintercept = boot_sample_variance_ci_bias[row, column][[1]][2]) ,col="red", lwd = 1)+
  geom_vline(aes(xintercept = lower_bound[row, column]) ,col="green", lwd = 1)+
  geom_vline(aes(xintercept = upper_bound[row, column]) ,col="green", lwd = 1)+
  scale_linetype_manual( guide = guide_legend())+
  xlab(TeX("$\\beta$"))+
  ylab("Frequency") +
  theme(text = element_text(size=14), legend.position = "none")

}

 grid.arrange(plot_boot_sample_ci(2,3),plot_boot_sample_ci(1,1))
```

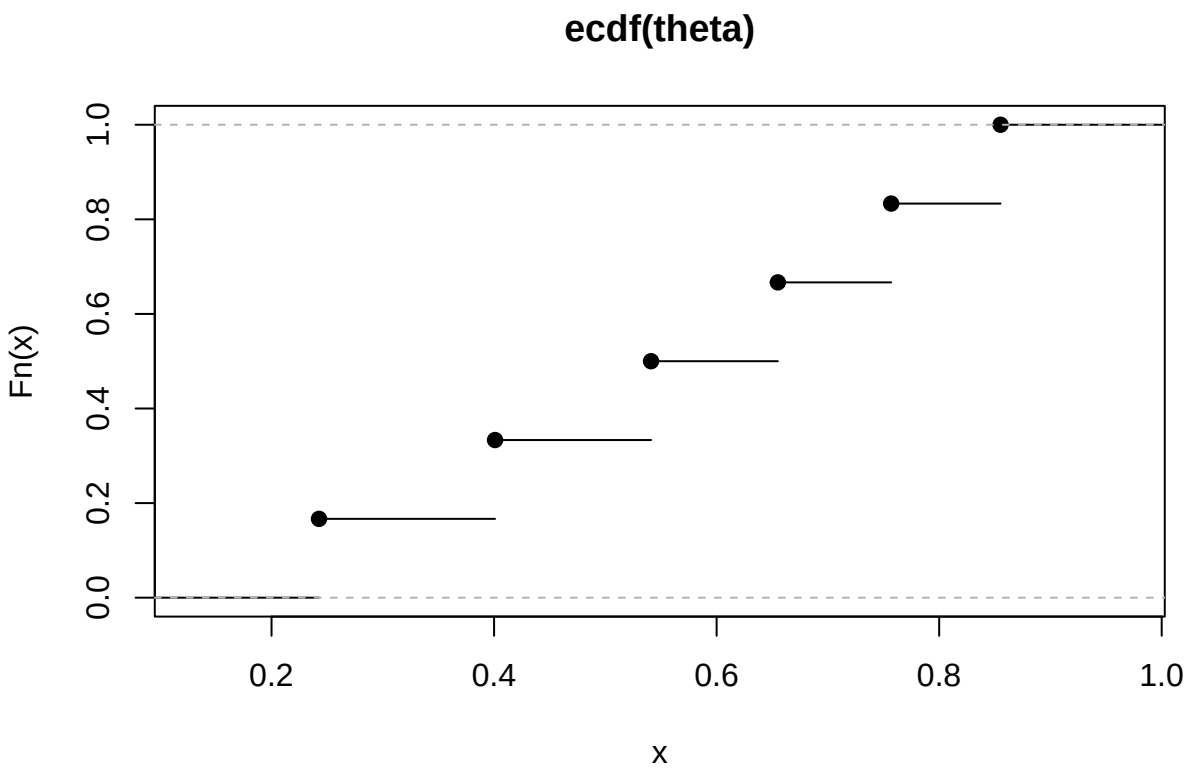


1.3 Point d)

```
#We know that these two values are equal
sum(eigen(sample_cov)$values) & sum(diag(sample_cov))
```

```
## [1] TRUE
```

```
j_star=4
theta_j_star=sum(eigen(sample_cov)$values[1:j])/sum(eigen(sample_cov)$values)

theta=numeric(6)
for( i in 1:length(theta)){
  theta[i]=sum(eigen(sample_cov)$values[1:i]/sum(eigen(sample_cov)$values))
}

plot(ecdf(theta))
```



1.4 Point e)

1.5 Point f)

Make things more feasible

```
set.seed(abs(636-555-3226))
ind <- sample(1:nrow(df),5000,FALSE)
sub_data <- df[ind,]
```

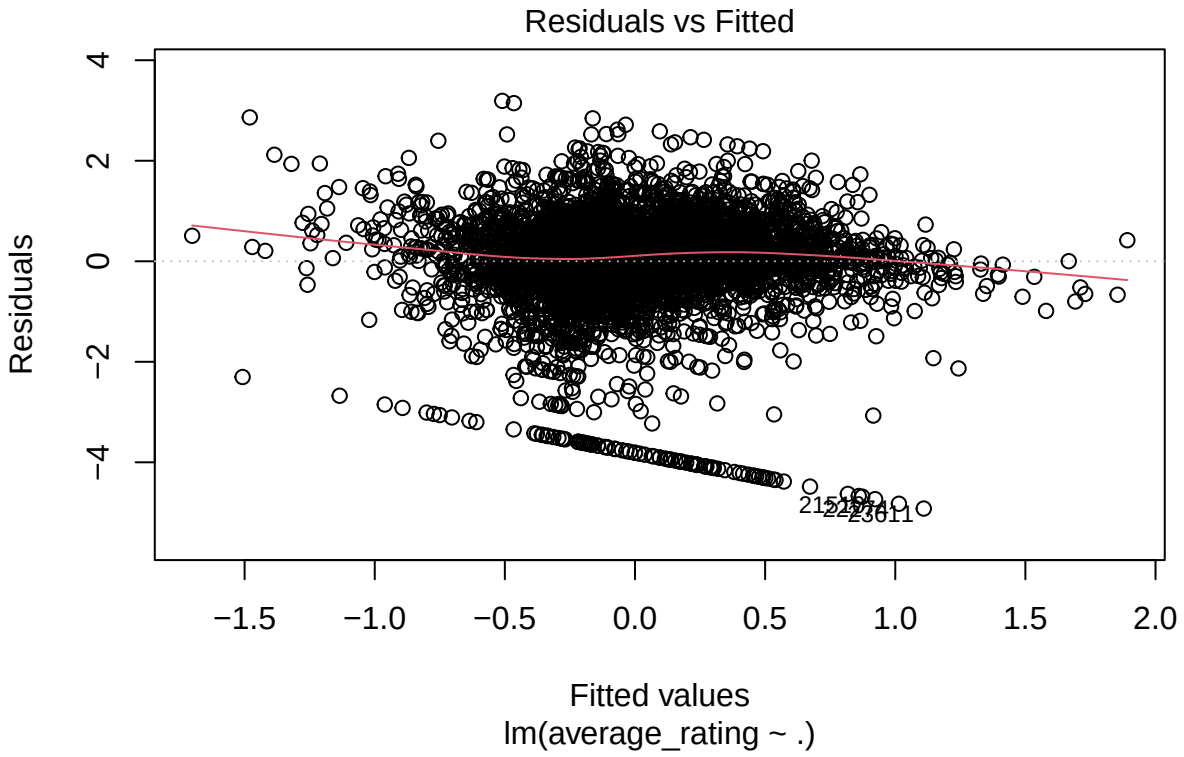
```
lm_game=lm(data=sub_data, average_rating ~ .)

res_sub_data=matrix(NA, 1, ncol(sub_data)+2)
res_sub_data[1,]=c(lm_game$coefficients, summary(lm_game)$r.squared,
               max((lm_game$coef["playingtime"]-lm_game$coef["minage"])/(lm_game$coef["minplayers"]+lm_g

colnames(res_sub_data)=c(colnames(sub_data), "R2" , "theta_hat")
res_sub_data[,"yearpublished"]-1
```

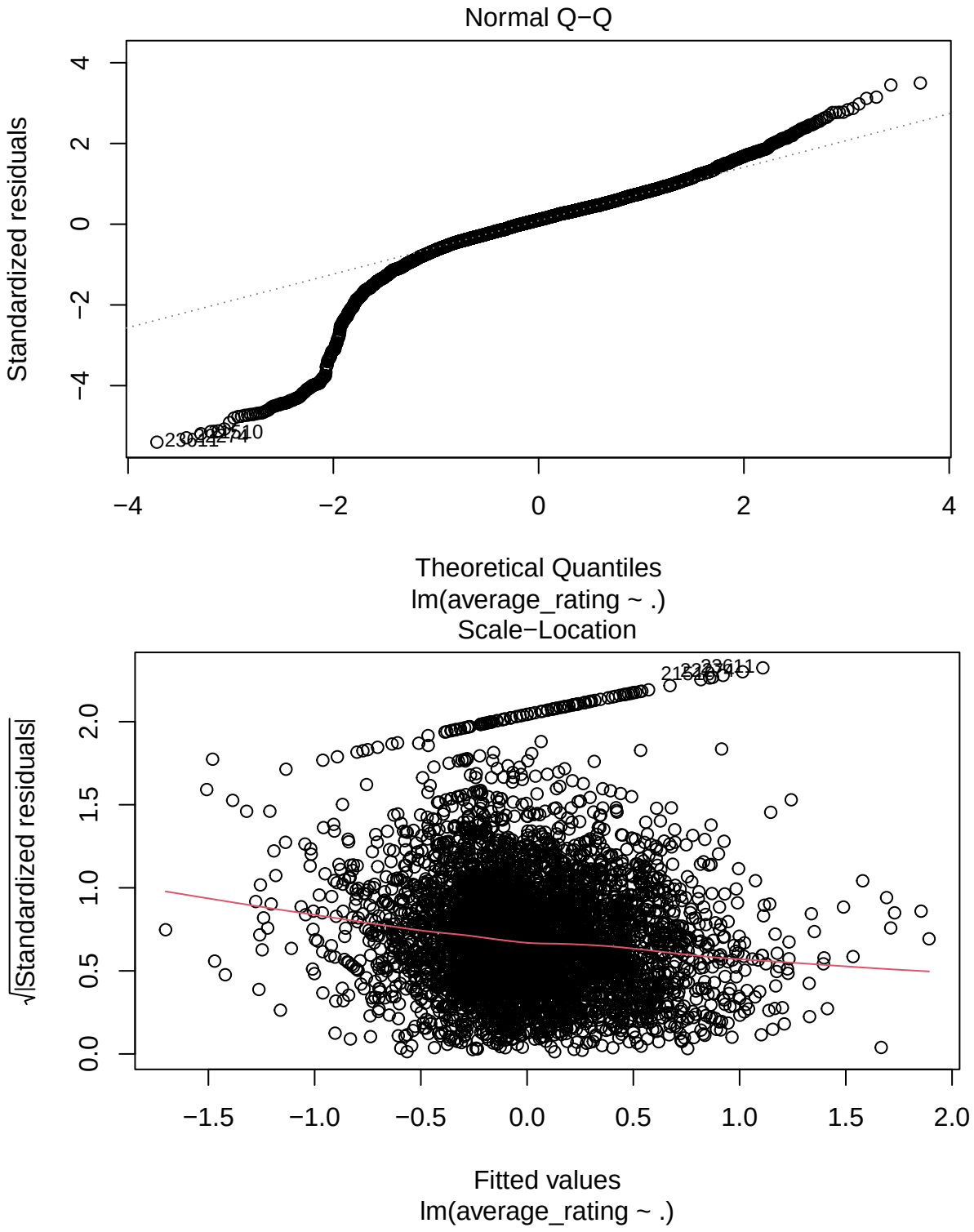
```
## yearpublished
##    -0.9916079
```

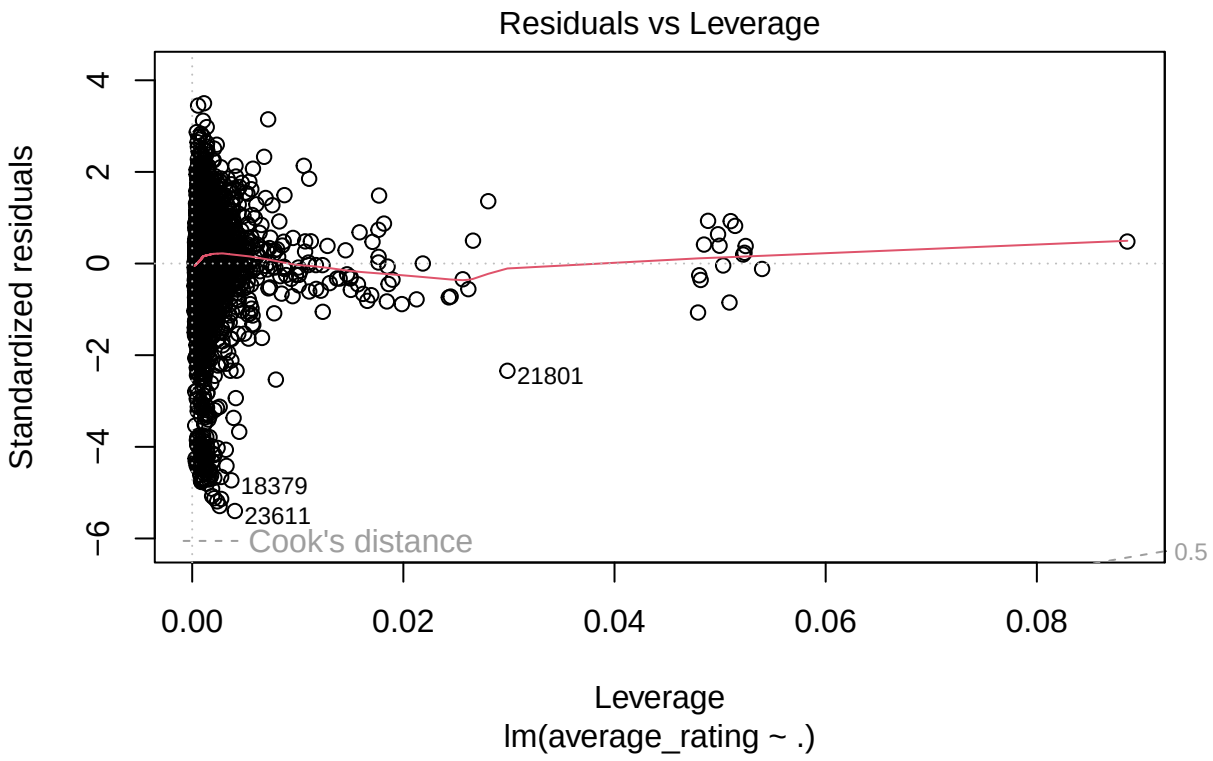
COMMENT

1.6 Point g)

```
plot(lm_game)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(average_rating ~ .)
Scale−Location
√|Standardized residuals|
Fitted values
lm(average_rating ~ .)



We can notice some features from this plots, useful for the choice of which bootstrap would be a sensitive choice. First of all the assumption of constant variance is not realistic in this case. Also the assumption of the errors distributed as a normal has problems, since we have a long tailed distribution. From this quick analysis of the model we can choose the bootstrap method more suited and explain why the others are not. 1. shuffle the errors -> this method relies on assumption that the chosen model fits the data well and the residuals have constant variance 2. Parametric bootstrap for logistic regression ->we are assuming that we know the distribution of the population which we don't 3. Resample the line of the two datasets -> this approach violates the assumption of constant (fixed) design matrix. however this applies perfectly to our case

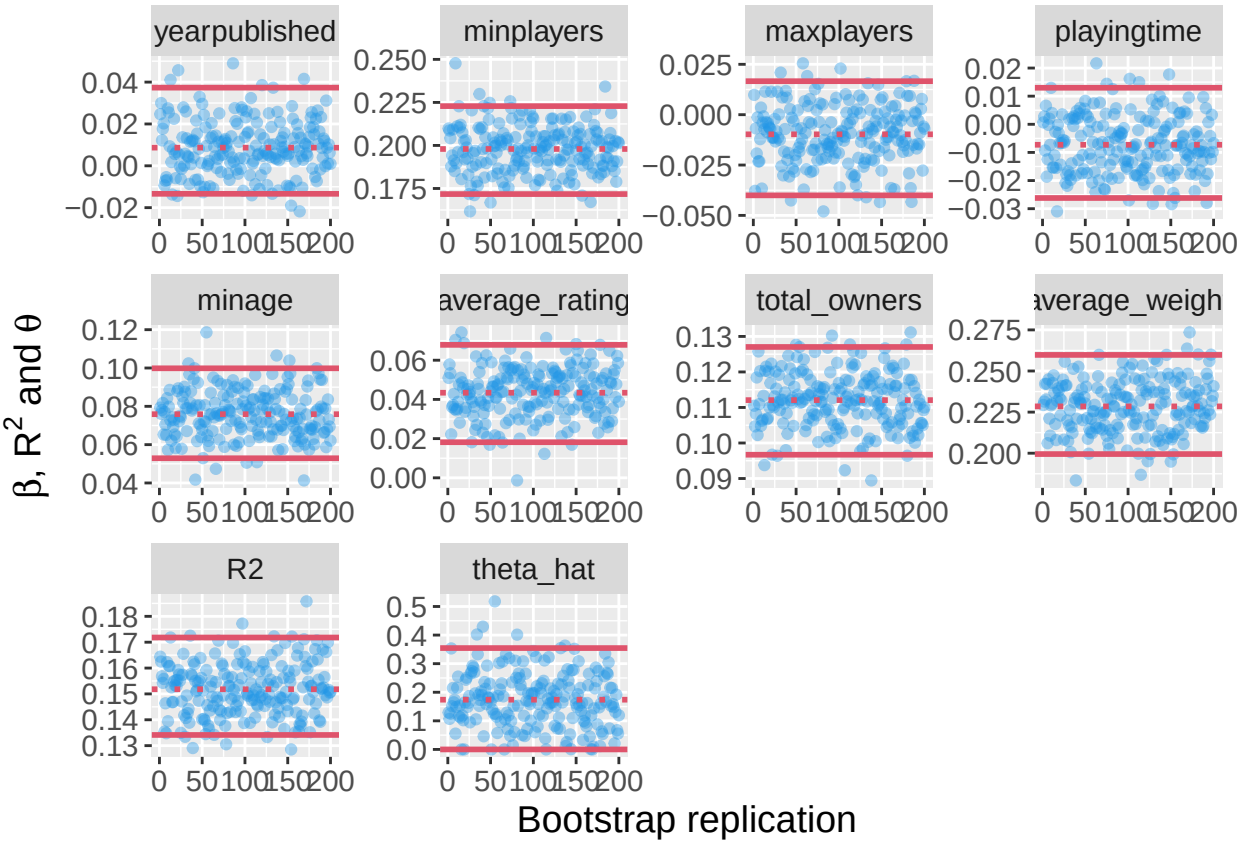
```
boot_reg_game <- function(data, B=200){

  n0 <- nrow(data)

  res <- replicate(B,{
    ind   <- sample(1:n0, n0, TRUE)
    Dstar <- data[ind,]
    Mstar <- lm(data=Dstar, average_rating ~ .)

    c(Mstar$coef, summary(Mstar)$r.squared, max((Mstar$coef["playingtime"]-Mstar$coef["minage"])/(Mstar$
  })

  t(res)
}

res_game_boot=boot_reg_game(data=sub_data)
colnames(res_game_boot)=c(colnames(sub_data), "R2" , "theta_hat")
```

Compute the estimates

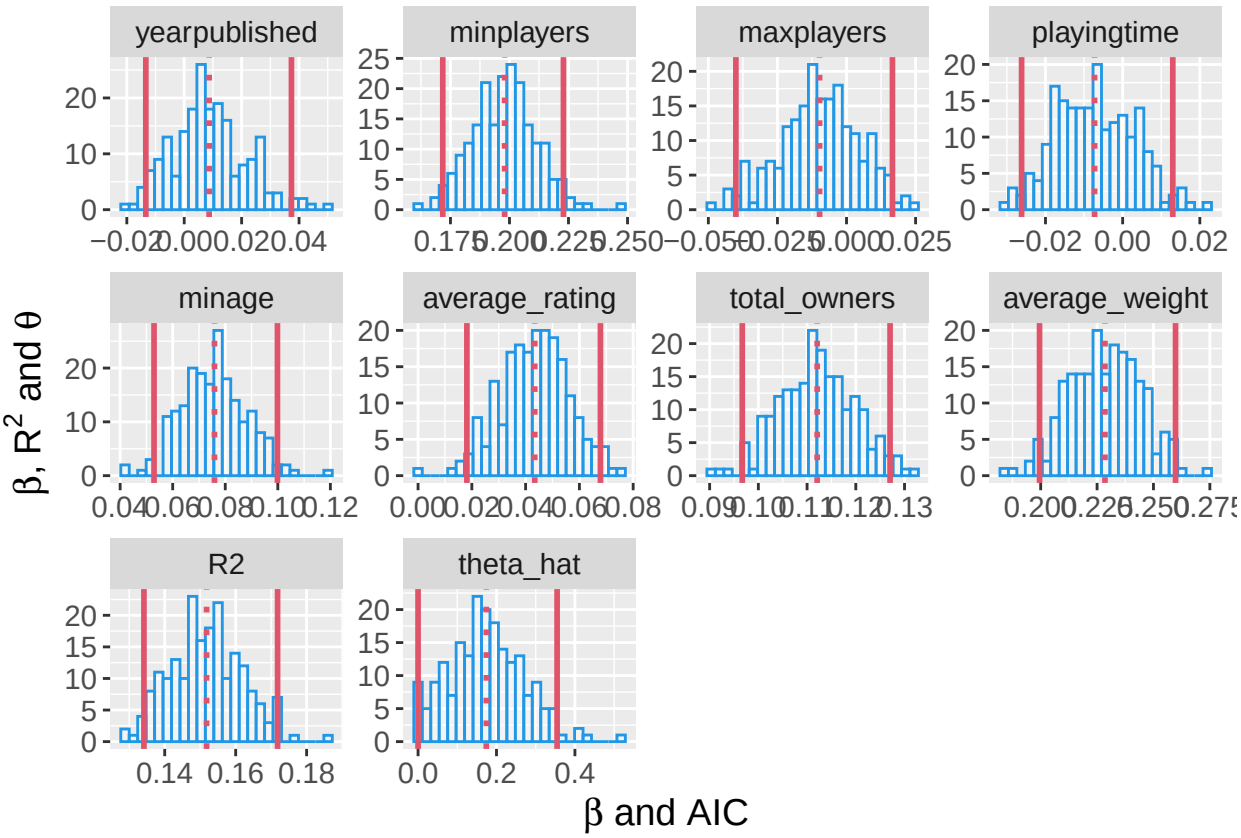
```
melted <- reshape2::melt(res_game_boot)
melted <- melted %>% group_by(Var2) %>% mutate( boot_sd = sd(value),
                                                boot_lowerbound = quantile(value, .025), boot_upperbound = quan
                                                boot_mean = mean(value), bias=boot_mean-res_sub_data[,as.chara
```

Plot the results

```
ggplot( melted )+
  geom_point(aes(x=Var1,y=value),alpha=.4,col=4)+
  geom_hline(aes(yintercept = boot_upperbound) ,col=2, lwd = 1)+
  geom_hline(aes(yintercept = boot_lowerbound) ,col=2, lwd = 1)+
  geom_hline(aes(yintercept = boot_mean) ,col=2, lwd = 1, lty=3)+
  facet_wrap(~Var2,scales = "free")+
  xlab("Bootstrap replication")+
  ylab(TeX("$\\beta$, $R^2$ and \\theta")) +
  theme(text = element_text(size=14), legend.position = "none")
```




```
ggplot( melted )+
  geom_histogram(aes(x=value),alpha=.4,col=4,fill="white",bins=25)+
  geom_vline(aes(xintercept = boot_upperbound) ,col=2, lwd = 1)+
  geom_vline(aes(xintercept = boot_lowerbound) ,col=2, lwd = 1)+
  geom_vline(aes(xintercept = boot_mean) ,col=2, lwd = 1, lty=3)+
  facet_wrap(~Var2,scales = "free")+
  xlab(TeX("$\\beta$ and AIC"))+
  ylab(TeX("$\\beta$, $R^2$ and \\theta")) +
  theme(text = element_text(size=14), legend.position = "none")
```



1.7 Point h)

Jackknife approach

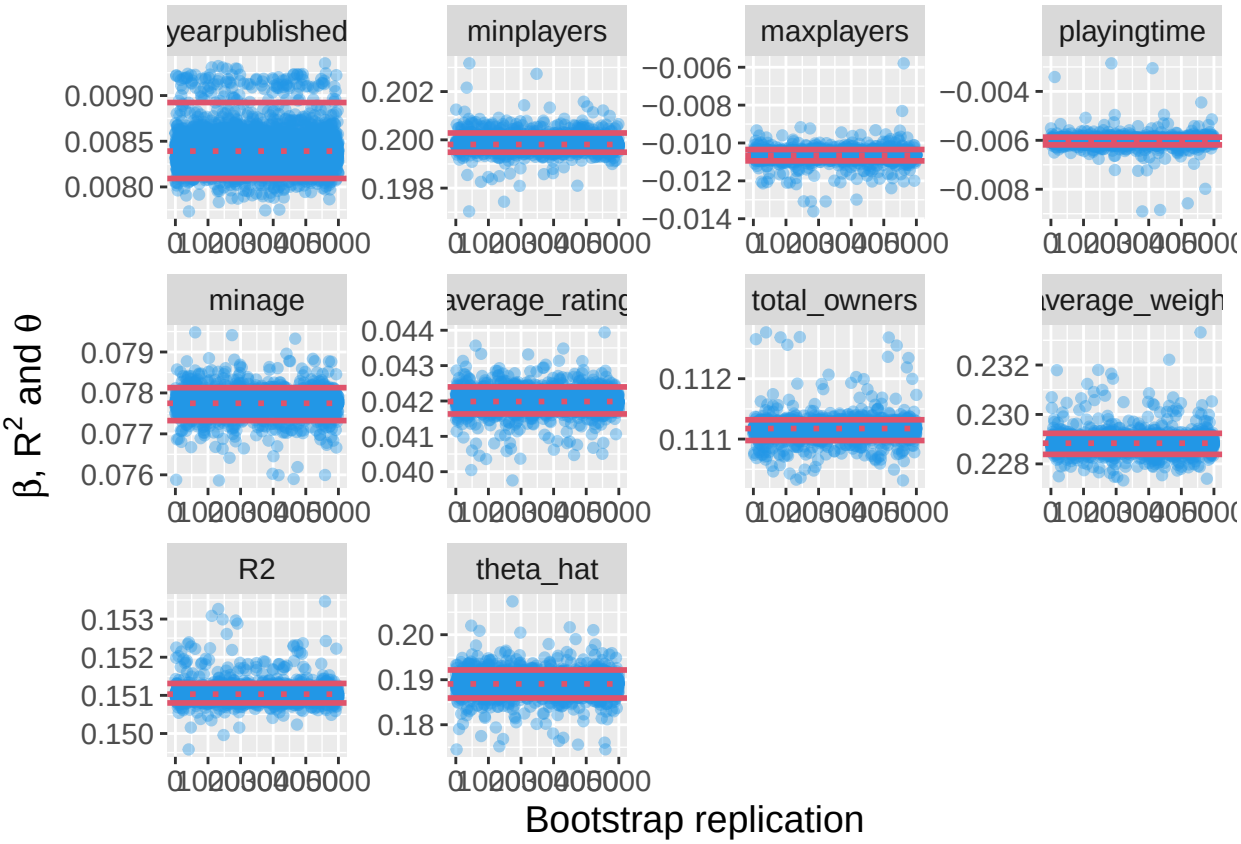
```
jack_reg_game = function(data, B){

  n0 <- nrow(data)

  res=matrix(NA, n0, ncol(data)+2)
for(i in 1:n0){
  Mjack=lm(data=data[-i,], average_rating ~ .)
  res[i,]=c(Mjack$coef, summary(Mjack)$r.squared, max((Mjack$coef["playingtime"]-Mjack$coef["minage"])/
}



  return(res)
}

res_game_jack=jack_reg_game(sub_data, 10)
colnames(res_game_jack)=c(colnames(sub_data), "R2" , "theta_hat")
```

Compute the jackknife estimate, bias-corrected estimate and standard error

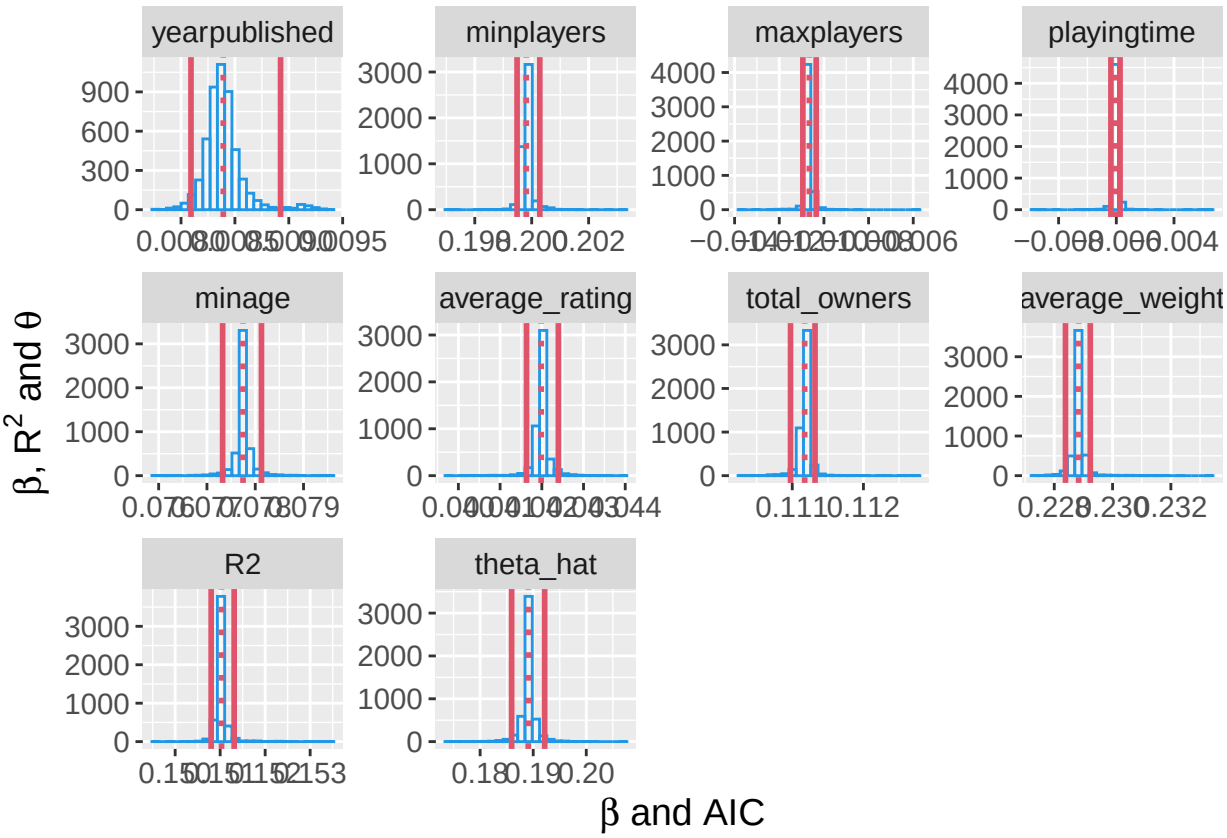
```
melted <- reshape2::melt(res_game_jack)
melted <- melted %>% group_by(Var2) %>% mutate( boot_sd = sd(value),
                                                boot_lowerbound = quantile(value, .025), boot_upperbound = quar
                                                boot_mean = mean(value), bias=boot_mean-res_sub_data[,as.chara
```

Plot the results # Exercise 2 - We need some music!

```
ggplot( melted )+
  geom_point(aes(x=Var1,y=value),alpha=.4,col=4)+
  geom_hline(aes(yintercept = boot_upperbound) ,col=2, lwd = 1)+
  geom_hline(aes(yintercept = boot_lowerbound) ,col=2, lwd = 1)+
  geom_hline(aes(yintercept = boot_mean) ,col=2, lwd = 1, lty=3)+
  facet_wrap(~Var2,scales = "free")+
  xlab("Bootstrap replication")+
  ylab(TeX("$\\beta$, $R^2$ and \\theta")) +
  theme(text = element_text(size=14), legend.position = "none")
```




```
ggplot( melted )+
  geom_histogram(aes(x=value),alpha=.4,col=4,fill="white",bins=25)+
  geom_vline(aes(xintercept = boot_upperbound) ,col=2, lwd = 1)+
  geom_vline(aes(xintercept = boot_lowerbound) ,col=2, lwd = 1)+
  geom_vline(aes(xintercept = boot_mean) ,col=2, lwd = 1, lty=3)+
  facet_wrap(~Var2,scales = "free")+
  xlab(TeX("$\\beta$ and AIC"))+
  ylab(TeX("$\\beta$, $R^2$ and \\theta")) +
  theme(text = element_text(size=14), legend.position = "none")
```



1.8 Point a)

1.9 Point b)

1.10 Point c)

1.11 Point d)

1.12 Point e)

1.13 Point f)

1.14 Point g)