EXERCISE 1

Miriam Mercuri 5207057

2023-06-06

EXERCISE 1 - RISIKO! is back, with friends

```
library(ggplot2)
library(ggcorrplot)
```

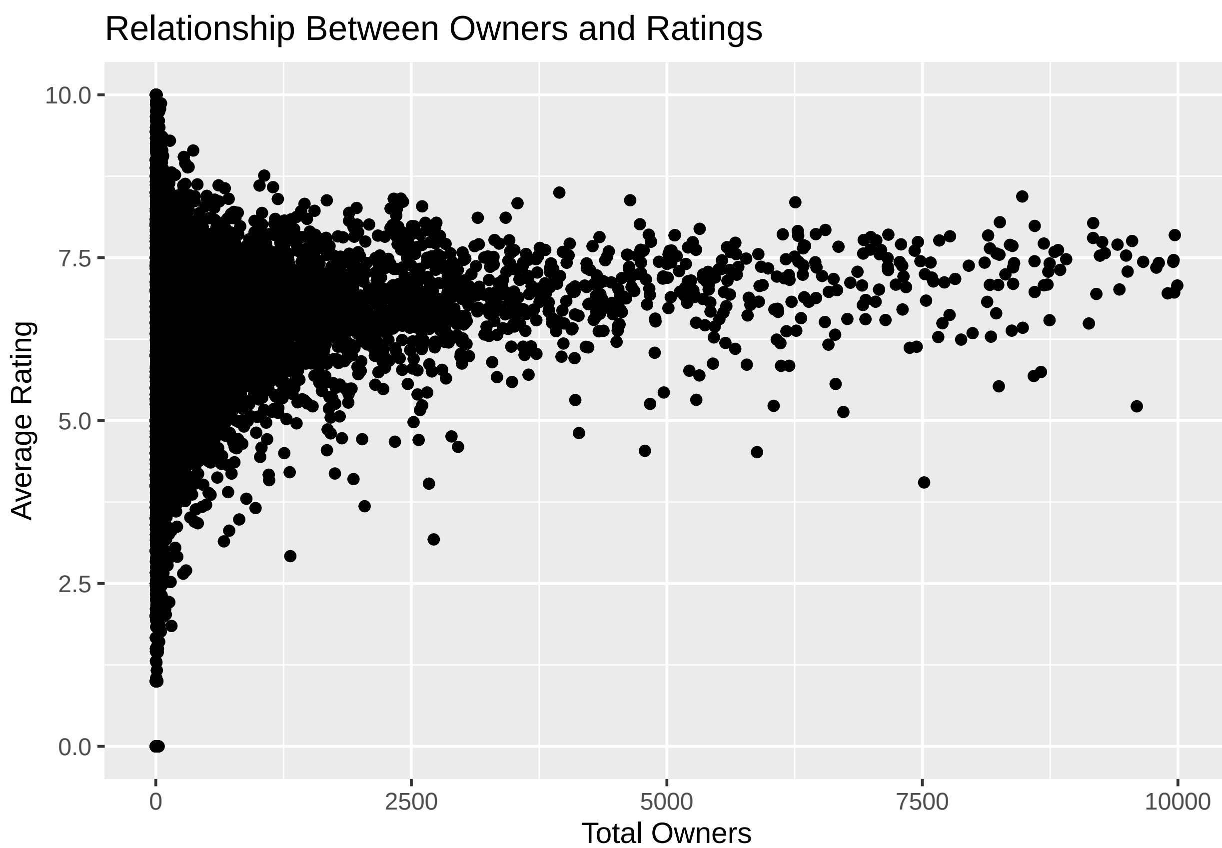
```
## Warning: il pacchetto 'ggcorrplot' è stato creato con R versione 4.2.3
```

Point a

```
games <- readRDS("games_preprocessed.RDS")
```

The dataset *game* contains data about around 25000 board games.The variables that describe each game are eight. They are the year in which it has been published (*yearpublished*), the minimum and the maximum number of players that can play the game (*minplayers*, *maxplayers*) and the minimun age to play (*minage*). Then there is the *average_rating*, *total_owners* and *average_weight*.All are quantitative variables except for the first (*id*) and second (*name*) ones that we are not going to consider for our exercise.

```
ggplot(games, aes(x = total_owners, y = average_rating)) +
  geom_point() +
  xlab("Total Owners") +
  ylab("Average Rating") +
  ggtitle("Relationship Between Owners and Ratings")
```

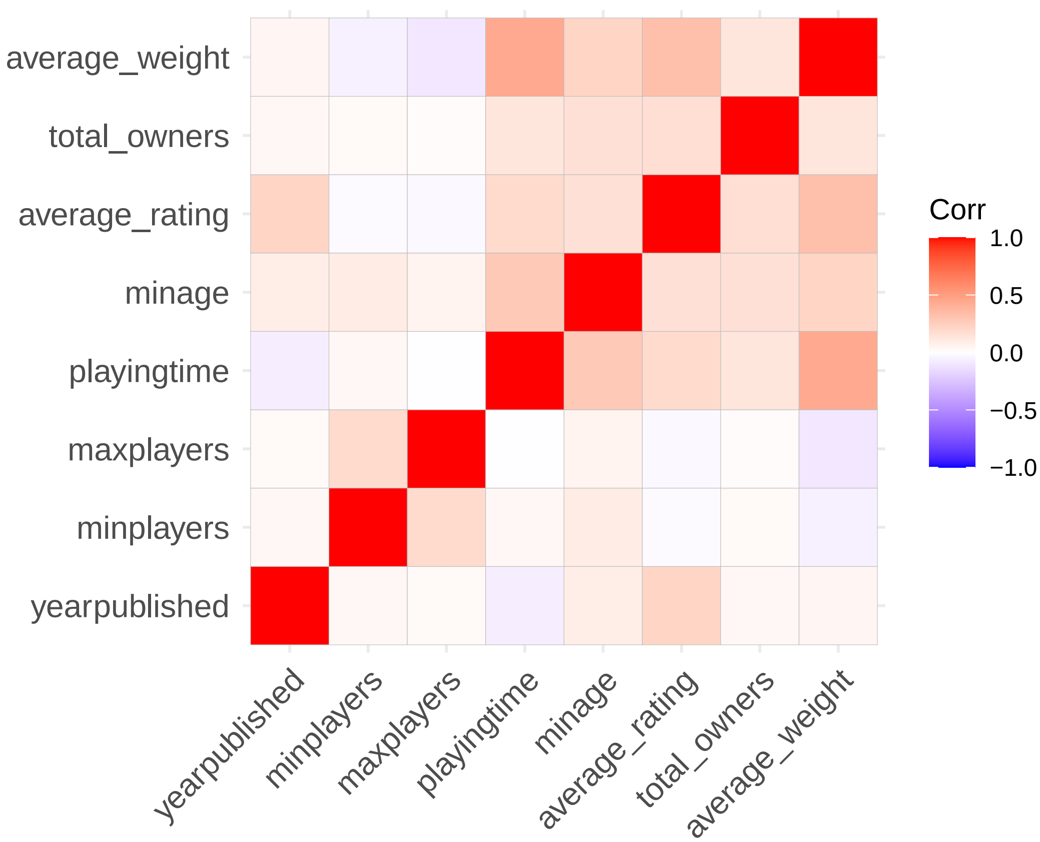


Considering, for example, the two variable *average_rating* and *total_owners*, the plot above make us see which is the relationship between them.

##Point b

```
games <- games[,-c(1,2)]
games <- scale(games)
```

```
cov_matrix <- cov(games)
ggcorrplot(cov_matrix)
```



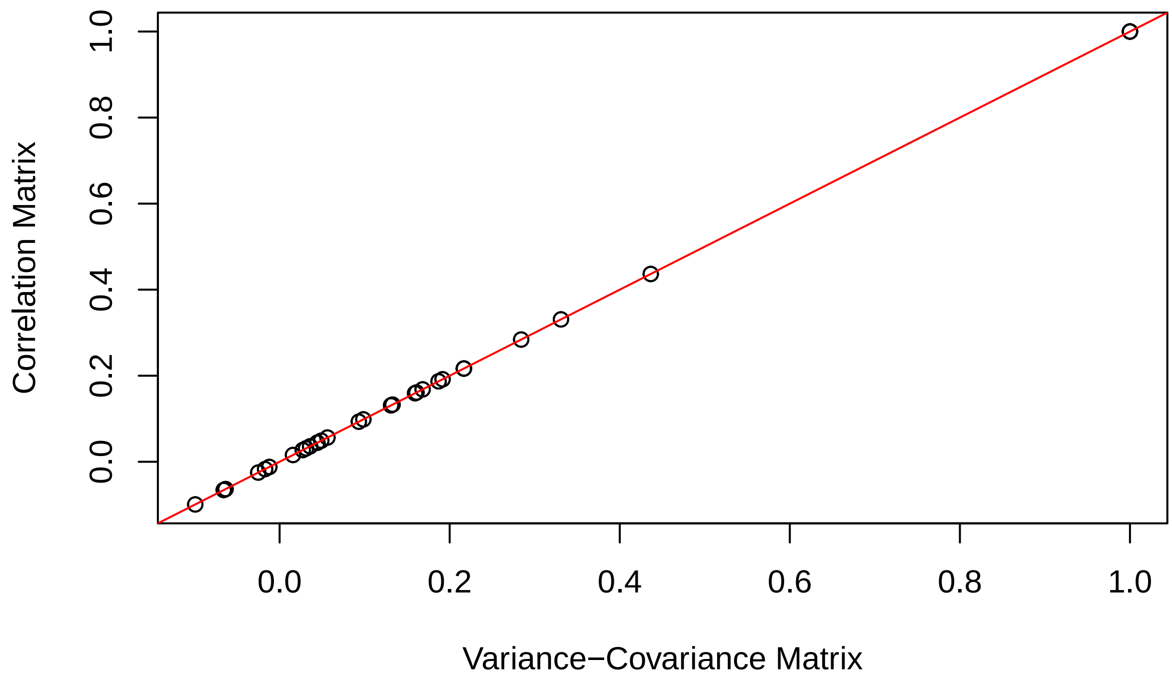

```r
cov_matrix
```

```
##                 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
```

From the plot and the variance-covariance matrix values we can see that the variables don't have a strong relationship in general.

```r
cor_matrix <- cor(games)
plot(cov_matrix, cor_matrix,
     xlab = "Variance-Covariance Matrix",
```

```
      ylab = "Correlation Matrix")
abline(0,1,col = "red")
```



The correlation matrix is the same as the variance-covariance matrix because our dataset has been scaled, resulting in equivalent covariance and correlation values.

Point c

Now we perform nonparametric bootstrap to obtain standard error estimates for each of the entries of the variance-covariance matrix.

```
set.seed(12345)

B <- 1000
boot <- function(B, games){

  results <- matrix(NA, nrow = B, ncol = ncol(games)^2)
  bootstrap <- replicate(B, {

    bootSample <- games[sample(nrow(games), replace = T),]
    bootCov <- cov(bootSample)
    i <- seq_len(B)
    results[i,] <- as.vector(bootCov)
  })
  return(bootstrap)
}
```

```r
boot <- boot(1000,games)

standard_errors <- apply(boot, 1, sd)
standard_errors
```

```
##  [1] 0.016767571 0.005726342 0.005392461 0.006026930 0.006611704 0.006527865
##  [7] 0.006366352 0.006391746 0.005726342 0.030878533 0.012807945 0.008199441
## [13] 0.007928323 0.006332489 0.007044940 0.006013313 0.005392461 0.012807945
## [19] 0.082934774 0.005253855 0.006605301 0.005991273 0.004863436 0.005935743
## [25] 0.006026930 0.008199441 0.005253855 0.012317292 0.006937599 0.006049556
## [31] 0.006927497 0.007735376 0.006611704 0.007928323 0.006605301 0.006937599
## [37] 0.018955155 0.007341887 0.004957443 0.007548761 0.006527865 0.006332489
## [43] 0.005991273 0.006049556 0.007341887 0.015166272 0.005447973 0.007162792
## [49] 0.006366352 0.007044940 0.004863436 0.006927497 0.004957443 0.005447973
## [55] 0.044025974 0.006719733 0.006391746 0.006013313 0.005935743 0.007735376
## [61] 0.007548761 0.007162792 0.006719733 0.010072221
```

Now we obtain confidence intervals of the bootstrap distribution using the percentile approach and the central limit theorem (CLT).

```r
means <- apply(boot, 1, mean)
qnt_CI <- t(apply(boot, 1, function(z) quantile(z, probs = c(0.025,0.975)))
)
clt_CI <- means + c(-1,1)*1.96*standard_errors

rbind(clt_CI, qnt_CI)
```

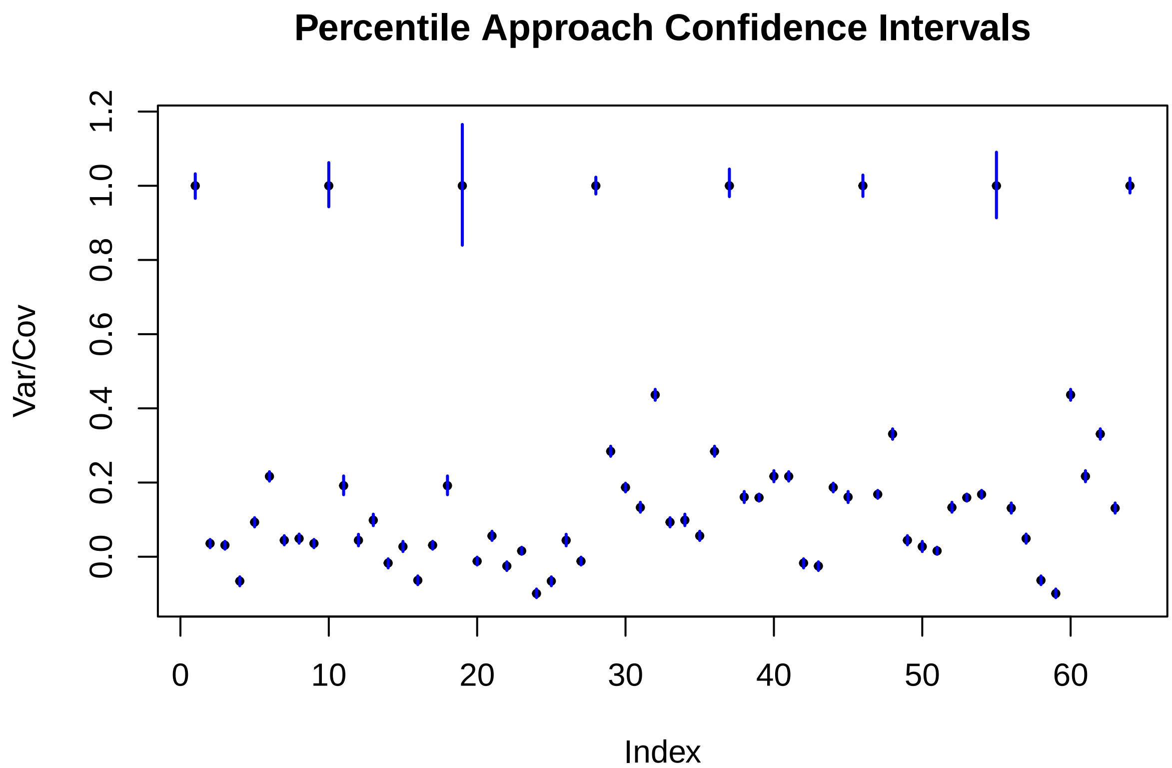
```
## Warning in rbind(clt_CI, qnt_CI): number of columns of result is not a multiple
## of vector length (arg 1)
```

```
##                2.5%        97.5%
## clt_CI  0.966718388  0.046801768
##         0.966018551  1.032263428
##         0.024402515  0.046665580
##         0.020378496  0.041586425
##        -0.078290232 -0.054121259
##         0.080447117  0.105306030
##         0.203889506  0.229230892
##         0.032090356  0.056857923
##         0.036403092  0.061183394
##         0.024402515  0.046665580
##         0.943256839  1.062456369
##         0.167140878  0.217810959
##         0.029444450  0.060590671
##         0.083838015  0.114793438
##        -0.030089711 -0.005514134
##         0.013955372  0.041418569
##        -0.075307029 -0.051694131
##         0.020378496  0.041586425
##         0.167140878  0.217810959
##         0.839705292  1.165336107
##        -0.022306759 -0.001048049
##         0.043989502  0.068808024
##        -0.037090523 -0.013674687
##         0.006647197  0.025644745
```

```
##        -0.110218691 -0.086922765
##        -0.078290232 -0.054121259
##         0.029444450  0.060590671
##        -0.022306759 -0.001048049
##         0.977715919  1.023158126
##         0.270827173  0.298012546
##         0.174780819  0.198311202
##         0.120069085  0.146786285
##         0.421806170  0.451254021
##         0.080447117  0.105306030
##         0.083838015  0.114793438
##         0.043989502  0.068808024
##         0.270827173  0.298012546
##         0.970719365  1.044995863
##         0.146129688  0.175978674
##         0.149929832  0.168983541
##         0.202165608  0.232001561
##         0.203889506  0.229230892
##        -0.030089711 -0.005514134
##        -0.037090523 -0.013674687
##         0.174780819  0.198311202
##         0.146129688  0.175978674
##         0.971154164  1.029002602
##         0.157487352  0.178896864
##         0.316628884  0.344774945
##         0.032090356  0.056857923
##         0.013955372  0.041418569
##         0.006647197  0.025644745
##         0.120069085  0.146786285
##         0.149929832  0.168983541
##         0.157487352  0.178896864
##         0.913547816  1.090524603
##         0.117541681  0.144938869
##         0.036403092  0.061183394
##        -0.075307029 -0.051694131
##        -0.110218691 -0.086922765
##         0.421806170  0.451254021
##         0.202165608  0.232001561
##         0.316628884  0.344774945
##         0.117541681  0.144938869
##         0.980555524  1.020535724
```

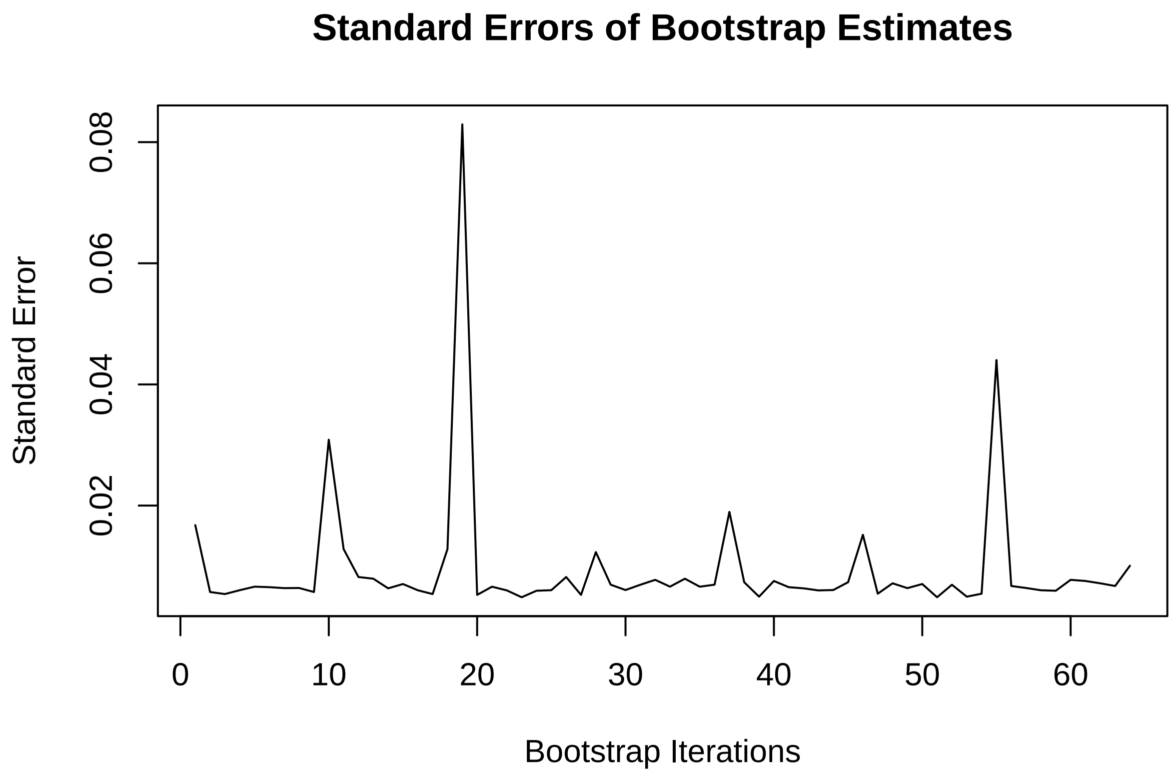
```r
cov_hat <- as.vector(cov_matrix)

x <- seq(length(cov_hat))
y <- cov_hat
y_upper <- unlist(qnt_CI[,2])
y_lower <- unlist(qnt_CI[,1])
plot(y, pch = 19, cex = 0.5,ylim = range(c(y_lower, y_upper)), ylab = "Var/Cov", xlab = "Index",
     main = "Percentile Approach Confidence Intervals")
segments(x, y_lower, x, y_upper, lwd = 1.5,col = "blue")
```



Above there is the plot of the confidence intervals of each entry of the variance-covariance matrix (Index) usign the percentile approach.

```
plot(1:length(standard_errors), standard_errors, type = "l",
     xlab = "Bootstrap Iterations", ylab = "Standard Error",
     main = "Standard Errors of Bootstrap Estimates")
```



Point d

Show the behavior of the values $\{\theta_j\}_{j=1}^{p}$ and estimate j^*.

```
eigen_values <- eigen(cov_matrix)$values
theta <- cumsum(eigen_values) / sum(eigen_values)
theta
```

```
## [1] 0.2426729 0.4008816 0.5410761 0.6550491 0.7568552 0.8551548 0.9359288
## [8] 1.0000000
```

```
jstar <- min(which(theta > 0.76))
jstar
```

```
## [1] 6
```

Point e

Performe bootstrap to estimate the bias and standard error of the estimators for the parameters in the sequence $\{\theta_j\}_{j=1}^{p}$.

```
boot_theta <- matrix(0, nrow = B, ncol = length(theta))
boot_jstar <- numeric(B)

for (i in 1:B) {

  boot_sample <- games[sample(nrow(games), replace= TRUE),]
```

```
  cov_boot <- cov(boot_sample)
  eigen_boot <- eigen(cov_boot)$values

  boot_theta[i,] <- cumsum(eigen_boot) / sum(eigen_boot)

  boot_jstar[i] <- min(which(boot_theta[i,] > 0.76))

}
```

```
bias <- colMeans(boot_theta)-theta
se <- apply(boot_theta, 2, sd)
bias
```

```
## [1] 2.660301e-04 8.058590e-04 7.381508e-04 9.166828e-04 1.767983e-03
## [6] 2.008943e-04 9.475436e-05 0.000000e+00
```

```
se
```

```
## [1] 0.003174574 0.002647125 0.002873812 0.003279194 0.002552820 0.002106016
## [7] 0.001101662 0.000000000
```

Also report the bootstrap estimates and standard errors for the estimator of j^*.

```
bias_jstar <- mean(boot_jstar) - jstar
se_jstar <- sd(boot_jstar)
bias_jstar
```

```
## [1] -0.278
```

```
se_jstar
```

```
## [1] 0.4482376
```

Finally, give an estimate for $P(j^* \hat{=} 5)$.

```
p_jstar <- mean(boot_jstar==5)
p_jstar
```

```
## [1] 0.278
```

Point f

Run a linear regression with average_rating as target variable, regressed over all the other variables.

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

sub_data <- as.data.frame(sub_data)
regression <- lm(sub_data$average_rating~., data = sub_data)
summary(regression)
```

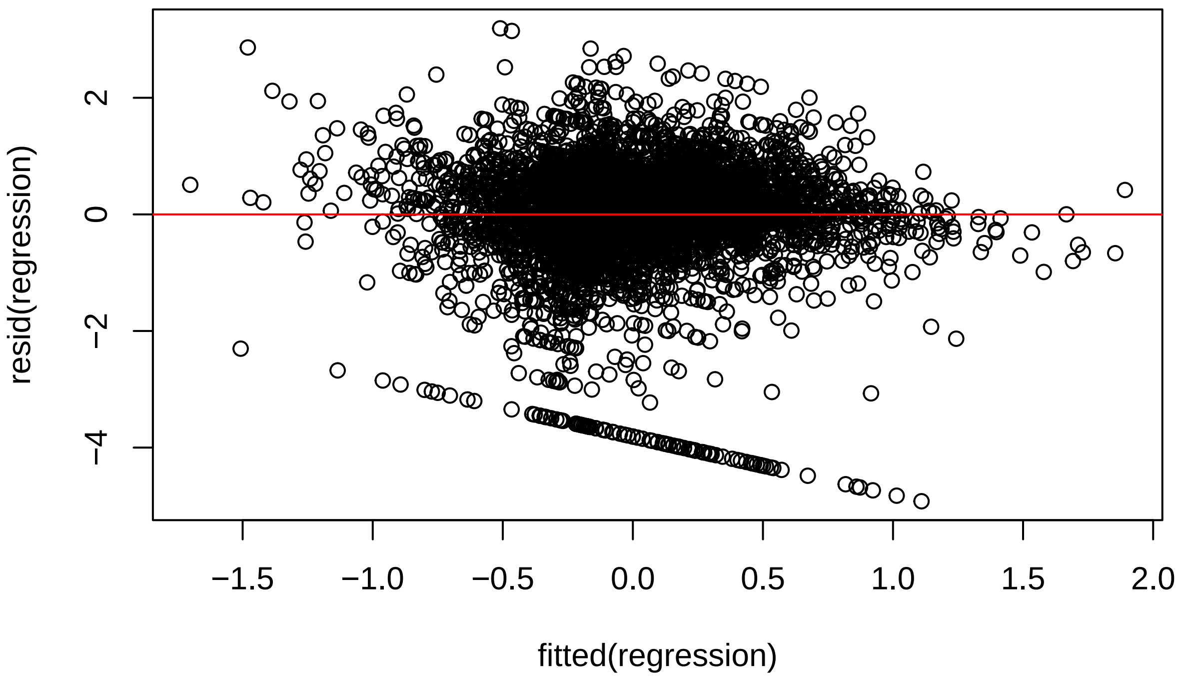
```
##
## Call:
## lm(formula = sub_data$average_rating ~ ., data = sub_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.9202 -0.3275  0.0989  0.4873  3.1911
```

```
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.008392   0.012916   0.650  0.51587
## yearpublished   0.199808   0.013454  14.851  < 2e-16 ***
## minplayers     -0.010653   0.013532  -0.787  0.43119
## maxplayers     -0.006019   0.013517  -0.445  0.65614
## playingtime     0.077747   0.014689   5.293 1.26e-07 ***
## minage          0.041984   0.013271   3.164  0.00157 **
## total_owners    0.111174   0.013423   8.283  < 2e-16 ***
## average_weight  0.228837   0.014440  15.848  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9129 on 4992 degrees of freedom
## Multiple R-squared:  0.151,  Adjusted R-squared:  0.1498
## F-statistic: 126.9 on 7 and 4992 DF,  p-value: < 2.2e-16
```

The intercept is equal to 0.008392. It corresponds to the expected average rate of a board game when the other variables are equal to zero. And so it is not interpretable in this problem because we are talking about the average rating of a board game that does not exist. We can use the R^2 to assess the goodness of fit of our linear regression model, that is equal to 0.151. It means that only 15,1% of the the variation in *average_rating* is explained by the regressors.

Point g

```r
plot(fitted(regression), resid(regression))
abline(0,0,col = "red")
```



Below we used the paired bootstrap method to provide bias, standard error and confidence intervals for regression coefficients, adjusted R^2 index and our $\hat{\theta}$. In particular, we chose this method and, for example, not the bootstrap of the errors because in the plot above it is possible to see that the residuals seem to not have constant variance.

```
set.seed(abs(636-555-3226))
perform_paired_bootstrap <- function(data, B) {
  num_coeffs <- length(coef(regression))
  bootstrap_estimates <- matrix(0, nrow = B, ncol = num_coeffs)

  for (i in 1:B) {
    bootstrap_indices <- sample(nrow(data), replace = TRUE)
    bootstrap_sample <- data[bootstrap_indices, ]
    bootstrap_regression <- lm(average_rating ~ ., data = bootstrap_sample)
    bootstrap_estimates[i, ] <- coef(bootstrap_regression)
  }

  coefficient_bias <- apply(bootstrap_estimates, 2, mean) - coef(regression)
  coefficient_std_error <- apply(bootstrap_estimates, 2, sd)
  coefficient_confidence_intervals <- t(apply(bootstrap_estimates, 2, function(x) quantile(x, c(0.025, 0

  adjusted_r_squared <- numeric(B)
  for (i in 1:B) {
    bootstrap_indices <- sample(nrow(data), replace = TRUE)
    bootstrap_sample <- data[bootstrap_indices, ]
    bootstrap_regression <- lm(average_rating ~ ., data = bootstrap_sample)
    adjusted_r_squared[i] <- 1 - (1 - summary(bootstrap_regression)$r.squared) * ((nrow(bootstrap_sample
```

```r
  }
  adjusted_r_squared_bias <- mean(adjusted_r_squared) - summary(regression)$adj.r.squared
  adjusted_r_squared_std_error <- sd(adjusted_r_squared)
  adjusted_r_squared_confidence_interval <- quantile(adjusted_r_squared, c(0.025, 0.975))

  theta_estimates <- numeric(B)
  coeff <- coef(regression)

  for (i in 1:B) {
    boot_sample <- data[sample(nrow(data), replace = TRUE), ]
    regression_boot <- lm(average_rating ~ ., data = boot_sample)
    coeff_boot <- coef(regression_boot)
    theta_boot <- max(((coeff_boot[5] - coeff_boot[6])/(coeff_boot[3]+coeff_boot[2])), 0)
    theta_estimates[i] <- theta_boot
  }

  bias <- mean(theta_estimates) - (coeff[5] - coeff[6])/(coeff[3]+coeff[2])
  se <- sd(theta_estimates)
  ci <- quantile(theta_estimates, probs = c(0.025, 0.975))

  results <- list(
    coefficient_bias = coefficient_bias,
    coefficient_std_error = coefficient_std_error,
    coefficient_confidence_intervals = coefficient_confidence_intervals,
    adjusted_r_squared_bias = adjusted_r_squared_bias,
    adjusted_r_squared_std_error = adjusted_r_squared_std_error,
    adjusted_r_squared_confidence_interval = adjusted_r_squared_confidence_interval,
    theta_bias = bias,
    theta_std_error = se,
    theta_confidence_interval = ci
  )

  return(results)
}

results <- perform_paired_bootstrap(sub_data, B)
results
```

```
## $coefficient_bias
##    (Intercept)  yearpublished     minplayers     maxplayers    playingtime
##   2.590343e-04  -5.313236e-04   5.391762e-05  -3.615397e-04  -2.715688e-04
##         minage   total_owners average_weight
##  -9.566880e-05   4.931990e-04  -6.421485e-05
##
## $coefficient_std_error
## [1] 0.012917019 0.014882237 0.014706065 0.009575085 0.013402492 0.013637539
## [7] 0.007677498 0.017932378
##
## $coefficient_confidence_intervals
##              2.5%      97.5%
## [1,] -0.01545676 0.03319964
## [2,]  0.17137553 0.22927323
## [3,] -0.03924184 0.01808103
## [4,] -0.02516037 0.01208000
```

```
## [5,]  0.05100863 0.10383718
## [6,]  0.01570397 0.06778159
## [7,]  0.09695728 0.12696910
## [8,]  0.19381726 0.26420035
##
## $adjusted_r_squared_bias
## [1] 0.0005191378
##
## $adjusted_r_squared_std_error
## [1] 0.01122473
##
## $adjusted_r_squared_confidence_interval
##      2.5%     97.5%
## 0.1292476 0.1717044
##
## $theta_bias
## playingtime
## 0.008709073
##
## $theta_std_error
## [1] 0.1101246
##
## $theta_confidence_interval
##      2.5%     97.5%
## 0.0000000 0.4221232
```

Point h

```r
set.seed(abs(636-555-3226))
perform_jackknife <- function(data) {
  num_coeffs <- length(coef(regression))
  num_samples <- nrow(data)
  jackknife_estimates <- matrix(0, nrow = num_samples, ncol = num_coeffs)

  for (i in 1:num_samples) {
    jackknife_sample <- data[-i, ]
    jackknife_regression <- lm(average_rating ~ ., data = jackknife_sample)
    jackknife_estimates[i, ] <- coef(jackknife_regression)
  }

  coefficient_bias <- num_samples * (colMeans(jackknife_estimates) - coef(regression)) / (num_samples -
  coefficient_std_error <- sqrt(num_samples * (colMeans(jackknife_estimates^2) - (colMeans(jackknife_es
  coefficient_confidence_intervals <- t(apply(jackknife_estimates, 2, function(x) quantile(x, c(0.025, 0

  adjusted_r_squared <- numeric(num_samples)
  for (i in 1:num_samples) {
    jackknife_sample <- data[-i, ]
    jackknife_regression <- lm(average_rating ~ ., data = jackknife_sample)
    adjusted_r_squared[i] <- 1 - (1 - summary(jackknife_regression)$r.squared) * ((nrow(jackknife_sampl
  }
  adjusted_r_squared_bias <- num_samples * (mean(adjusted_r_squared) - summary(regression)$adj.r.square
  adjusted_r_squared_std_error <- sqrt(num_samples * (mean(adjusted_r_squared^2) - (mean(adjusted_r_squa
  adjusted_r_squared_confidence_interval <- quantile(adjusted_r_squared, c(0.025, 0.975))
```

```
  theta_estimates <- numeric(num_samples)
  coeff <- coef(regression)

  for (i in 1:num_samples) {
    jackknife_sample <- data[-i, ]
    jackknife_regression <- lm(average_rating ~ ., data = jackknife_sample)
    coeff_boot <- coef(jackknife_regression)
    theta_estimates[i] <- max(((coeff_boot[5] - coeff_boot[6])/(coeff_boot[3] + coeff_boot[2])), 0)
  }

  theta_bias <- num_samples * (mean(theta_estimates) - (coeff[5] - coeff[6])/(coeff[3] + coeff[2])) / (n
  theta_std_error <- sqrt(num_samples * (mean(theta_estimates^2) - (mean(theta_estimates))^2) / (num_sam
  theta_confidence_interval <- quantile(theta_estimates, probs = c(0.025, 0.975))

  results <- list(
    coefficient_bias = coefficient_bias,
    coefficient_std_error = coefficient_std_error,
    coefficient_confidence_intervals = coefficient_confidence_intervals,
    adjusted_r_squared_bias = adjusted_r_squared_bias,
    adjusted_r_squared_std_error = adjusted_r_squared_std_error,
    adjusted_r_squared_confidence_interval = adjusted_r_squared_confidence_interval,
    theta_bias = theta_bias,
    theta_std_error = theta_std_error,
    theta_confidence_interval = theta_confidence_interval
  )

  return(results)
}
results_jackknife <- perform_jackknife(sub_data)
results_jackknife
```

```
## $coefficient_bias
##    (Intercept)  yearpublished     minplayers     maxplayers    playingtime
##  -1.045900e-08   3.160400e-08   7.927530e-09  -6.638226e-08   7.077107e-10
##         minage   total_owners average_weight
##  -2.011886e-08   7.649813e-08  -7.538168e-09
##
## $coefficient_std_error
## [1] 0.0001819372 0.0002094197 0.0001985008 0.0001395934 0.0001921217
## [6] 0.0001934172 0.0001108170 0.0002526914
##
## $coefficient_confidence_intervals
##              2.5%        97.5%
## [1,]  0.008092459  0.008923529
## [2,]  0.199486341  0.200284337
## [3,] -0.010942538 -0.010345240
## [4,] -0.006186247 -0.005870274
## [5,]  0.077324374  0.078128852
## [6,]  0.041632962  0.042396845
## [7,]  0.110976154  0.111320283
## [8,]  0.228385290  0.229235412
##
## $adjusted_r_squared_bias
```

```
## [1] -0.0001703989
##
## $adjusted_r_squared_std_error
## [1] 0.0001616389
##
## $adjusted_r_squared_confidence_interval
##      2.5%     97.5%
## 0.1494391 0.1499510
##
## $theta_bias
##  playingtime
## 1.791877e-07
##
## $theta_std_error
## [1] 0.00155067
##
## $theta_confidence_interval
##      2.5%     97.5%
## 0.1859348 0.1921601
```

The differences between the results of the paired bootstrap and the jackknife methods can be attributed to their different resampling techniques. The paired bootstrap randomly samples with replacement, while the jackknife systematically leaves out observations. They also have different statistical properties. The paired bootstrap provides an estimate of the distribution of the coefficients, while the jackknife estimates the variance of the coefficients. This leads to variations in the results.