Stealingbaseballsim

2024-06-19

```
library(readr)
June24 <- read_csv("June24.csv")
```

```
## Rows: 20 Columns: 21
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): Player, GS-GP, SBATT
## dbl (18): Number, OPS, AVG, AB, R, H, 2B, 3B, HR, RBI, SLG, BB, HBP, SO, GDP...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
View(June24)
```

```
stealcleaning <- function(df){
  df[c("Steals", "StealAttempts")] <- str_split_fixed(df$SBATT, ',', 2)
  df$SBATT <- as.integer(df$SBATT)
  df$Steals <- as.integer(df$Steals)
  df$StealAttempts <- as.integer(df$StealAttempts)
  df <- df %>% select(-SBATT) %>% mutate("Stealper" = Steals/StealAttempts)
  df}
june240 <- stealcleaning(June24)
```

```
## Warning in stealcleaning(June24): NAs introduced by coercion
```

```
su <- june240$Steals %>% sum() / june240$StealAttempts %>% sum()
```

```
lineup618 <- c("Ethan Gibson",
               "Ryan Wynn",
               "Colby Wallace",
               "Tommy Roldan",
               "Johnnie Lopez",
               "David Wiley",
               "Nick Parham",
               "Cale Stricklin",
               "Sam Mummau")
lineup616 <- c("Sam Mummau",
               "Ryan Wynn",
               "Tommy Roldan",
               "Colby Wallace",
               "Johnnie Lopez",
               "David Wiley",
```

```
               "James Taussig",
               "Nick Parham",
               "Cale Stricklin")
lineup621 <- c("Ethan Gibson",
               "Cade Belyeu",
               "Tommy Roldan",
               "Robbie Lavey",
               "Ryan Wynn",
               "Johnnie Lopez",
               "David Wiley",
               "Nick Parham",
               "Cale Stricklin")
```

Creating a Batter Probability Matrix:

```
battermatrix <- function(df,lineup){
  df <- df %>% rename("Doubles" = "2B", "Triples" = "3B")
  df <- df %>% mutate("PA" = AB+BB+HBP+SF+SH)
  df <- df %>% mutate("singleper" = (H - (Doubles + Triples + HR))/ PA) %>%   mutate("doubleper" = (Doub
  #df$ROEper[5] <- 0
  #df$ROEper[9] <- 0
  df$ROEper[4] <- 0
  df$ROEper[17] <- 0
  #df$ROEper[16] <- 0
  df$ROEper[3] <- 0
  df$ROEper[6] <- 0
  df$ROEper[7] <- 0
  batterprobs <- data.frame()
  batter1 <- df %>% filter(Player == lineup[1]) %>% select("Kper","Outper","singleper","doubleper","trip
  batter2 <- df %>% filter(Player == lineup[2]) %>% select("Kper","Outper","singleper","doubleper","trip
  batter3 <- df %>% filter(Player == lineup[3]) %>% select("Kper","Outper","singleper","doubleper","trip
  batter4 <- df %>% filter(Player == lineup[4]) %>% select("Kper","Outper","singleper","doubleper","trip
  batter5 <- df %>% filter(Player == lineup[5]) %>% select("Kper","Outper","singleper","doubleper","trip
  batter6 <- df %>% filter(Player == lineup[6]) %>% select("Kper","Outper","singleper","doubleper","trip
  batter7 <- df %>% filter(Player == lineup[7]) %>% select("Kper","Outper","singleper","doubleper","trip
  batter8 <- df %>% filter(Player == lineup[8]) %>% select("Kper","Outper","singleper","doubleper","trip
  batter9 <- df %>% filter(Player == lineup[9]) %>% select("Kper","Outper","singleper","doubleper","trip
  batterprobs <- data.frame(batter1,batter2,batter3,batter4,batter5,batter6,batter7,batter8,batter9)
  batterprobs}
batterprobs <- battermatrix(June24,lineup621)
batterprobs
```

```
##               batter1    batter2    batter3      batter4    batter5    batter6
## Kper      0.202531646 0.11363636 0.16000000 0.1836734694 0.25000000 0.25925926
## Outper    0.335468354 0.27318182 0.38700000 0.2855102041 0.36553846 0.27304938
## singleper 0.113924051 0.09090909 0.16000000 0.1836734694 0.17307692 0.18518519
## doubleper 0.050632911 0.04545455 0.01333333 0.0612244898 0.07692308 0.03703704
## tripleper 0.012658228 0.02272727 0.01333333 0.0000000000 0.00000000 0.00000000
## HRper     0.012658228 0.11363636 0.01333333 0.0816326531 0.01923077 0.03703704
## walkper   0.265822785 0.27272727 0.21333333 0.1632653061 0.07692308 0.17283951
## ROEper    0.006303797 0.00000000 0.00000000 0.0002040816 0.00000000 0.01090123
## DPper     0.000000000 0.00000000 0.01333333 0.0408163265 0.01923077 0.00000000
## SFper     0.000000000 0.06818182 0.02666667 0.0000000000 0.01923077 0.02469136
```

```
##           batter7   batter8    batter9
## Kper        0.175 0.2333333 0.15789474
## Outper      0.375 0.4336667 0.29856140
## singleper   0.175 0.1333333 0.19298246
## doubleper   0.025 0.0000000 0.01754386
## tripleper   0.000 0.0000000 0.00000000
## HRper       0.025 0.0000000 0.00000000
## walkper     0.200 0.2000000 0.31578947
## ROEper      0.000 0.0000000 0.00000000
## DPper       0.025 0.0000000 0.01754386
## SFper       0.000 0.0000000 0.00000000
```

The function battermatrix() takes two inputs being a dataframe of the universal batting statistics, and the lineup. The lineup is simply a vector of the players' names in the order of which they hit. This function then spits out a probability matrix for each of the 9 batters, which will be used as averages for the baseball simulation.

List of Assumptions:

1. Errors: we have calculated errors to result in a ROE to occur when a hit nor an out are recorded, for some players this happens 0% of the time but I will originally use their individual ROE probability. **2.** Lineup: The current lineup is just something I've put together based on these players college stats and positions. With just creating a batter probability matrix, we could use this code for any lineup in baseball. **3.** 1st to 3rd: When a single is hit with a man on 1st we are assuming the general 70/30 rule (70% of the time the runner is content with staying at 2nd). Also a 2.5% chance of being thrown out when attempting to advance to 3rd.

As a sidenote, for all the probability matricies I calculate for the different scenarios, value of 0 means out, 1 means held up, 2 means advanced as far as possible.

firstthird is the probability data frame with my outcomes and corresponding probabilities

ft is coded to be the outcome of **1** random draw

```
v <- c(0,1,2)
p <- c(0.025,0.7,0.275)
firstthird <- data.frame(v,p,row.names=c("Out","2nd","3rd"))
ft <- firstthird[sample(nrow(firstthird), 1, replace = TRUE, prob = firstthird$p), -ncol(firstthird)]
```

4. 1st to Home: When a double is hit there are a variety of options when a runner leaves from 1st, they can hold up at 3rd, or go home and risk getting thrown out. (I have approximated the successfulness of scoring is 0.42, 0.04 for getting thrown out somewhere along the line, and 0.54 for holding up at third)

firsthome is the probability data frame with my outcomes and corresponding probabilities

fh is coded to be the outcome of **1** random draw

```
va <- c(0,1,2)
pr <- c(0.04,0.54,0.42)
firsthome <- data.frame(va,pr,row.names=c("Out","3rd","Home"))
fh <- firsthome[sample(nrow(firsthome), 1, replace = TRUE, prob = firsthome$pr), -ncol(firsthome)]
```

5. Base Movement with Errors: As the vast majority of the errors in the Valley Baseball League are made in the infield, a runner from first will have the option to try to reach 3rd base on an error or be content with pulling into 2nd. I have placed about a 75% chance that the runner will go all the way to 3rd as with an error the defense is usually unable to prevent runners movement.

errorthird is the probability data frame with my outcomes and corresponding probabilities

et is coded to be the outcome of **1** random draw

```
probs <- c(0.25,0.75)
val <- c(0,1)
errorthird <- data.frame(val,probs,row.names = c("Second","Third"))
et <- errorthird[sample(nrow(errorthird), 1, replace = TRUE, prob = errorthird$probs), -ncol(errorthird
```

6. 2nd to Home: When a Single is hit and there is a runner on 2nd, I have placed the odds of 0.58 of being safe at home 0.02 of being thrown out at home and 0.4 of stopping at third.

secondhome is the probability data frame with my outcomes and corresponding probabilities

sh is coded to be the outcome of **1** random draw

```
pro <- c(0.02,0.4,0.58)
secondhome <- data.frame(v,pro,row.names = c("Out","Third","Home"))
sh <- secondhome[sample(nrow(secondhome), 1, replace = TRUE, prob = secondhome$pro), -ncol(secondhome)]
```

7. Tagout probability: A portion of the average flyballs and groundballs totals that we have will result in a play at the plate we have that 3% of all these outs will result in a double play of a runner being thrown out at the plate when you start with a man on third.

tagout is the probability data frame with my outcomes and corresponding probabilities

to is coded to be the outcome of **1** random draw

```
probss <- c(0.03,0.97)
tagout <- data.frame(val,probss,row.names=c("Out","Safe"))
to <- tagout[sample(nrow(tagout), 1, replace = TRUE, prob = tagout$probss), -ncol(tagout)]
```

7. Double Plays: We have the data for how often the members of the team ground into double plays, but the uncertainty lies in which bases the defense will target when you have man on 1st and 2nd or bases loaded. With man on 1st and 2nd we have that 85% of the time runners at 2nd and 1st will be out. In bases loaded it is 67% on 2nd and 1st, 25% for home and 1st, and 8% for 3rd and 1st. (These are all just my estimates as I was having a difficult time finding an answer online)

doubleplay is the probability data frame with my outcomes and corresponding probabilities

dp is coded to be the outcome of **1** random draw

newdoubleplay is the probability data frame with my outcomes and corresponding probabilities

nd is coded to be the outcome of **1** random draw

```
probsss <- c(0.85,0.15)
doubleplay <- data.frame(val,probsss,row.names=c("Third","Second"))
dp <- doubleplay[sample(nrow(doubleplay), 1, replace = TRUE, prob = doubleplay$probsss), -ncol(doublepla
```

```
prob0 <- c(0.67,0.08,0.25)
newdoubleplay <- data.frame(v,prob0,row.names=c("Second","Third","Home"))
nd <- newdoubleplay[sample(nrow(newdoubleplay), 1, replace = TRUE, prob = newdoubleplay$prob0), -ncol(n
```

8. Another Note: When I discussed the First to Third, First to Home, and Second to Home probabilities, I also used the same distribution for multiple runners on base like 1st and 2nd, 2nd and 3rd, 1st and 3rd, and bases loaded. This may not be totally accurate but this entire game is just a simulation and I can always edit the numbers in the future if they prove to be unreliable.

Stealing Matrix:

```
stealmatrix <- function(stealrate,successrate){
  firstst <- 1.15*stealrate
  first <- c(firstst,successrate)
  secondst <- 0.7*stealrate
  second <- c(secondst,successrate)
  thirdst <- 0.01*stealrate
  thirdsu <- 0.3*successrate
  third <- c(thirdst,thirdsu)
  firstandsecondsu <- 1.05*successrate
  firstandsecond <- c(stealrate,firstandsecondsu)
  firstandthirdst <- 0.85*stealrate
  firstandthirdsu <- 1.1*successrate
  firstandthird <- c(firstandthirdst,firstandthirdsu)
  stealprobs <- data.frame(first,second,third,firstandsecond,firstandthird)
  stealprobs
}
stealmatrix(0.41,su)
```

```
##      first   second     third firstandsecond firstandthird
## 1 0.471500 0.287000 0.0041000      0.4100000     0.3485000
## 2 0.893617 0.893617 0.2680851      0.9382979     0.9829787
```

```
Value <- c(0,1,2,3,4,5,6,7,8,9)
```

```
bases <- c(0,0,0)
outs <- c(0)
runs <- c(0)
```

```
nobodyon <- function(prob){
  tau <- data.frame(Value,prob,row.names=c("Strikeout","Other Out","Single","Double","Triple","Home Run
  ab <- tau[sample(nrow(tau), 1, replace = TRUE, prob = tau$prob), -ncol(tau)]
  if (ab == 0|ab == 1|ab == 8|ab == 9){
  bases <- c(0,0,0)
  outs <- outs + 1
}
  if (ab == 2|ab == 6|ab == 7){
  bases <- c(1,0,0)
```

```
}
  if (ab == 3){
  bases <- c(0,1,0)
}
  if (ab == 4){
  bases <- c(0,0,1)
}
  if (ab == 5){
  bases <- c(0,0,0)
  runs <- runs + 1
}
  score <- c(bases,outs,runs)
  score
}
manonfirst <- function(prob){
  tau <- data.frame(Value,prob,row.names=c("Strikeout","Other Out","Single","Double","Triple","Home Run
  ab1 <- tau[sample(nrow(tau), 1, replace = TRUE, prob = tau$prob), -ncol(tau)]
  if (ab1 == 0|ab1 == 1|ab1 == 9){
  bases <- c(1,0,0)
  outs <- outs + 1
}
  if (ab1 == 6|ab1 == 7){
  bases <- c(1,1,0)
}
  if (ab1 == 2 & ft == 0){
  bases <- c(1,0,0)
  outs <- outs + 1
}
  if (ab1 == 2 & ft == 1){
  bases <- c(1,1,0)
}
  if (ab1 == 2 & ft == 2){
  bases <- c(1,0,1)
}
  if (ab1 == 3 & fh == 0){
  bases <- c(0,1,0)
  outs <- outs + 1
}
  if (ab1 == 3 & fh == 1){
  bases <- c(0,1,1)
}
  if (ab1 == 3 & fh == 2){
  bases <- c(0,1,0)
  runs <- runs + 1
}
  if (ab1 == 4){
  bases <- c(0,0,1)
  runs <- runs + 1
}
  if (ab1 == 5){
  bases <- c(0,0,0)
  runs <- runs + 2
}
```

```
  if (ab1 == 8){
  bases <- c(0,0,0)
  outs <- outs + 2
  }
  score <- c(bases,outs,runs)
  score
}
manonsecond <- function(prob){
  tau <- data.frame(Value,prob,row.names=c("Strikeout","Other Out","Single","Double","Triple","Home Run
  ab2 <- tau[sample(nrow(tau), 1, replace = TRUE, prob = tau$prob), -ncol(tau)]
  if (ab2 == 0|ab2 == 1|ab2 == 8|ab2 == 9){
  bases <- c(0,1,0)
  outs <- outs + 1
}
  if (ab2 == 2 & sh == 0){
  bases <- c(1,0,0)
  outs <- outs + 1
}
  if (ab2 == 2 & sh == 1){
  bases <- c(1,0,1)
}
  if (ab2 == 2 & sh == 2){
  bases <- c(1,0,0)
  runs <- runs + 1
}
  if (ab2 == 3){
  bases <- c(0,1,0)
  runs <- runs + 1
}
  if (ab2 == 4){
  bases <- c(0,0,1)
  runs <- runs + 1
}
  if (ab2 == 5){
  bases <- c(0,0,0)
  runs <- runs + 2
}
  if (ab2 == 6){
  bases <- c(1,1,0)
}
  if (ab2 == 7 & et == 0){
  bases <- c(1,1,0)
}
  if (ab2 == 7 & et == 1){
  bases <- c(1,0,1)
}
  score <- c(bases,outs,runs)
  score
}
manonthird <- function(prob){
   tau <- data.frame(Value,prob,row.names=c("Strikeout","Other Out","Single","Double","Triple","Home Ru
  ab3 <- tau[sample(nrow(tau), 1, replace = TRUE, prob = tau$prob), -ncol(tau)]
  if (ab3 == 1 & to == 0){
```

```
  bases <- c(0,0,0)
  outs <- outs + 2
}
  if (ab3 == 1 & to == 1){
  bases <- c(0,0,1)
  outs <- outs + 1
}
  if (ab3 == 0|ab3 == 8){
  bases <- c(0,0,1)
  outs <- outs + 1
}
  if (ab3 == 2|ab3 == 7){
  bases <- c(1,0,0)
  runs <- runs + 1
}
  if (ab3 == 3){
  bases <- c(0,1,0)
  runs <- runs + 1
}
  if (ab3 == 4){
  bases <- c(0,0,1)
  runs <- runs + 1
}
  if (ab3 == 5){
  bases <- c(0,0,0)
  runs <- runs + 2
}
  if (ab3 == 6){
  bases <- c(1,0,1)
}
  if (ab3 == 9){
  bases <- c(0,0,0)
  outs <- outs + 1
  runs <- runs + 1
  }
  score <- c(bases,outs,runs)
  score
}
manonfirstandsecond <- function(prob){
  tau <- data.frame(Value,prob,row.names=c("Strikeout","Other Out","Single","Double","Triple","Home Run
  ab4 <- tau[sample(nrow(tau), 1, replace = TRUE, prob = tau$prob), -ncol(tau)]
  if (ab4 == 0 | ab4 == 1 | ab4 == 9){
  bases <- c(1,1,0)
  outs <- outs + 1
}
  if (ab4 == 2 & sh == 0 & ft == 0){
  bases <- c(1,1,0)
  outs <- outs + 1
}
  if (ab4 == 2 & sh == 0 & ft == 1){
  bases <- c(1,1,0)
  outs <- outs + 1
}
```

```
  if (ab4 == 2 & sh == 0 & ft == 2){
  bases <- c(1,0,1)
  outs <- outs + 1
}
  if (ab4 == 2 & sh == 1 & (ft == 0 | ft == 2)){
  bases <- c(1,1,1)
}
  if (ab4 == 2 & sh == 1 & ft == 1){
  bases <- c(1,1,1)
}
  if (ab4 == 2 & sh == 2 & ft == 1){
  bases <- c(1,1,0)
  runs <- runs + 1
}
  if (ab4 == 2 & sh == 2 & ft == 0){
  bases <- c(1,0,0)
  runs <- runs + 1
  outs <- outs + 1
}
  if (ab4 == 2 & sh == 2 & ft == 2){
  bases <- c(1,0,1)
  runs <- runs + 1
}
  if (ab4 == 3 & fh == 0){
  bases <- c(0,1,0)
  runs <- runs + 1
  outs <- outs + 1
}
  if (ab4 == 3 & fh == 1){
  bases <- c(0,1,1)
  runs <- runs + 1
}
  if (ab4 == 3 & fh == 2){
  bases <- c(0,1,0)
  runs <- runs + 2
}
  if (ab4 == 4){
  bases <- c(0,0,1)
  runs <- runs + 2
}
  if (ab4 == 5){
  bases <- c(0,0,0)
  runs <- runs + 3
}
  if (ab4 == 6|ab4 == 7){
  bases <- c(1,1,1)
}
  if (ab4 == 8 & dp == 0){
  bases <- c(0,0,1)
  outs <- outs + 2
}
  if (ab4 == 8 & dp == 1){
  bases <- c(0,1,0)
```

```
  outs <- outs + 2
  }
  score <- c(bases,outs,runs)
  score
}
manonsecondandthird <- function(prob){
  tau <- data.frame(Value,prob,row.names=c("Strikeout","Other Out","Single","Double","Triple","Home Run
  ab5 <- tau[sample(nrow(tau), 1, replace = TRUE, prob = tau$prob), -ncol(tau)]
  if (ab5 == 1 & to == 0){
  bases <- c(0,1,0)
  outs <- outs + 2
}
  if (ab5 == 1 & to == 1){
  bases <- c(0,1,1)
  outs <- outs + 1
}
  if (ab5 == 0 | ab5 == 8){
  bases <- c(0,1,1)
  outs <- outs + 1
}
  if (ab5 == 2 & sh == 0){
  bases <- c(1,0,0)
  runs <- runs + 1
  outs <- outs + 1
}
  if (ab5 == 2 & sh == 1){
  bases <- c(1,0,1)
  runs <- runs + 1
}
  if (ab5 == 2 & sh == 2){
  bases <- c(1,0,0)
  runs <- runs + 2
}
  if (ab5 == 3){
  bases <- c(0,1,0)
  runs <- runs + 2
}
  if (ab5 == 4){
  bases <- c(0,0,1)
  runs <- runs + 2
}
  if (ab5 == 5){
  bases <- c(0,0,0)
  runs <- runs + 3
}
  if (ab5 == 6|(ab5 == 7 & et == 0)){
  bases <- c(1,1,1)
}
  if (ab5 == 7 & et == 1){
  bases <- c(1,0,1)
  runs <- runs + 1
}
  if (ab5 == 9){
```

```
  bases <- c(0,1,0)
  outs <- outs + 1
  runs <- runs + 1
  }
  score <- c(bases,outs,runs)
  score
}
basesloaded <- function(prob){
   tau <- data.frame(Value,prob,row.names=c("Strikeout","Other Out","Single","Double","Triple","Home Ru
  ab6 <- tau[sample(nrow(tau), 1, replace = TRUE, prob = tau$prob), -ncol(tau)]
  if (ab6 == 0){
  bases <- c(1,1,1)
  outs <- outs + 1
}
  if (ab6 == 1 & to == 0){
  bases <- c(1,1,0)
  outs <- outs + 2
}
  if (ab6 == 1 & to == 1){
  bases <- c(1,1,1)
  outs <- outs + 1
}
  if (ab6 == 2 & sh == 0 & (ft == 1 | ft == 0)){
  bases <- c(1,1,0)
  runs <- runs + 1
  outs <- outs + 1
}
  if (ab6 == 2 & sh == 0 & ft == 2){
  bases <- c(1,0,1)
  runs <- runs + 1
  outs <- outs + 1
}
  if (ab6 == 2 & sh == 1){
  bases <- c(1,1,1)
  runs <- runs + 1
}
  if (ab6 == 2 & sh == 2 & ft == 0){
  bases <- c(1,0,0)
  runs <- runs + 2
  outs <- outs + 1
}
  if (ab6 == 2 & sh == 2 & ft == 1){
  bases <- c(1,1,0)
  runs <- runs + 2
}
  if (ab6 == 2 & sh == 2 & ft == 2){
  bases <- c(1,0,1)
  runs <- runs + 2
}
  if (ab6 == 3 & fh == 0){
  bases <- c(0,1,0)
  runs <- runs + 2
  outs <- outs + 1
```

```
}
  if (ab6 == 3 & fh == 1){
  bases <- c(0,1,1)
  runs <- runs + 2
}
  if (ab6 == 3 & fh == 2){
  bases <- c(0,1,0)
  runs <- runs + 3
}
  if (ab6 == 4){
  bases <- c(0,0,1)
  runs <- runs + 3
}
  if (ab6 == 5){
  bases <- c(0,0,0)
  runs <- runs + 4
}
  if (ab6 == 6 | ab6 == 7){
  bases <- c(1,1,1)
  runs <- runs + 1
}
  if (ab6 == 8 & nd == 0){
  bases <- c(0,0,1)
  outs <- outs + 2
  runs <- runs + 1
}
  if (ab6 == 8 & nd == 1){
  bases <- c(0,1,0)
  outs <- outs + 2
  runs <- runs + 1
}
  if (ab6 == 8 & nd == 2){
  bases <- c(0,1,1)
  outs <- outs + 2
}
  if (ab6 == 9){
  bases <- c(1,1,0)
  outs <- outs + 1
  runs <- runs + 1
}
  score <- c(bases,outs,runs)
  score
}
manonfirstandthird <- function(prob){
  tau <- data.frame(Value,prob,row.names=c("Strikeout","Other Out","Single","Double","Triple","Home Run
  ab7 <- tau[sample(nrow(tau), 1, replace = TRUE, prob = tau$prob), -ncol(tau)]
  if (ab7 == 0){
  bases <- c(1,0,1)
  outs <- outs + 1
}
  if (ab7 == 1 & to == 0){
  bases <- c(1,0,0)
  outs <- outs + 2
```

```
}
  if (ab7 == 1 & to == 1){
  bases <- c(1,0,1)
  outs <- outs + 1
}
  if (ab7 == 2 & ft == 0){
  bases <- c(1,0,0)
  runs <- runs + 1
  outs <- outs + 1
}
  if (ab7 == 2 & ft == 1){
  bases <- c(1,1,0)
  runs <- runs + 1
}
  if (ab7 == 2 & ft == 2){
  bases <- c(1,0,1)
  runs <- runs + 1
}
  if (ab7 == 3 & fh == 0){
  bases <- c(0,1,0)
  runs <- runs + 1
  outs <- outs + 1
}
  if (ab7 == 3 & fh == 1){
  bases <- c(0,1,1)
  runs <- runs + 1
}
  if (ab7 == 3 & fh == 2){
  bases <- c(0,1,0)
  runs <- runs + 2
}
  if (ab7 == 4){
  bases <- c(0,0,1)
  runs <- runs + 2
}
  if (ab7 == 5){
  bases <- c(0,0,0)
  runs <- runs + 3
}
  if (ab7 == 6){
  bases <- c(1,1,1)
}
  if (ab7 == 7 & et == 0){
  bases <- c(1,1,1)
}
  if (ab7 == 7 & et == 1){
  bases <- c(1,1,0)
  runs <- runs + 1
}
  if (ab7 == 8){
  bases <- c(0,0,0)
  outs <- outs + 2
  runs <- runs + 1
```

```
}
  if (ab7 == 9){
  bases <- c(1,0,0)
  outs <- outs + 1
  runs <- runs + 1
  }
  score <- c(bases,outs,runs)
  score
}
```

```
score <- c(0,0,0,0,0)
```

```
v1 <- c(0,1,2)
      p1 <- c((stealmatrix(0.1,0.9)[1,1])*(1-(stealmatrix(0.1,0.9)[2,1])),1-stealmatrix(0.1,0.9)[1,1],(s
      firststeal <- data.frame(v1,p1,row.names=c("Out","1st","2nd"))
      fs <- firststeal[sample(nrow(firststeal), 1, replace = TRUE, prob = firststeal$p1), -ncol(firstst
firststeal
```

```
##     v1     p1
## Out  0 0.0115
## 1st  1 0.8850
## 2nd  2 0.1035
```

```
singlebatter <- function(og_score,i,s){
    if (og_score[1] == 0 & og_score[2] == 0 & og_score[3] == 0){
      trial0 <- nobodyon(batterprobs[,i])
      score <- c(og_score[1:3] != trial0[1:3], og_score[4:5] + trial0[4:5])
      }
    if (og_score[1] == 1 & og_score[2] == 0 & og_score[3] == 0){
      v1 <- c(0,1,2)
      p1 <- c((s[1,1])*(1-(s[2,1])),1-s[1,1],(s[1,1]*s[2,1]))
      firststeal <- data.frame(v1,p1,row.names=c("Out","1st","2nd"))
      fs <- firststeal[sample(nrow(firststeal), 1, replace = TRUE, prob = firststeal$p1), -ncol(firstst
      if (fs == 0){
        og_score <- c(0,0,0,og_score[4]+1,og_score[5])
        trial0 <- nobodyon(batterprobs[,i])
        score <- c(og_score[1:3] != trial0[1:3], og_score[4:5] + trial0[4:5])
      }
      if (fs == 2){
        og_score <- c(0,1,0,og_score[4],og_score[5])
      }
      if (fs == 1){
        trial1 <- manonfirst(batterprobs[,i])
        score <- c(og_score[1] == trial1[1], og_score[2] != trial1[2], og_score[3] != trial1[3], og_sco
      }}
      if (og_score[1] == 0 & og_score[2] == 1 & og_score[3] == 0){
        v2 <- c(0,1,2)
        p2 <- c((s[1,2])*(1-(s[2,2])),1-s[1,2],(s[1,2]*s[2,2]))
        secondsteal <- data.frame(v2,p2,row.names=c("Out","2nd","3rd"))
        ss <- secondsteal[sample(nrow(secondsteal), 1, replace = TRUE, prob = secondsteal$p2), -ncol(se
        if (ss == 0){
          og_score <- c(0,0,0,og_score[4]+1,og_score[5])
```

```
      trial0 <- nobodyon(batterprobs[,i])
      score <- c(og_score[1:3] != trial0[1:3], og_score[4:5] + trial0[4:5])
    }
    if (ss == 2){
      og_score <- c(0,0,1,og_score[4],og_score[5])
    }
    if (ss == 1){
      trial2 <- manonsecond(batterprobs[,i])
      score <- c(og_score[1] != trial2[1], og_score[2] == trial2[2], og_score[3] != trial2[3], og_s
    }
  }
    if (og_score[1] == 0 & og_score[2] == 0 & og_score[3] == 1){
      v3 <- c(0,1,2)
      p3 <- c((s[1,3])*(1-(s[2,3])),1-s[1,3],(s[1,3]*s[2,3]))
      thirdsteal <- data.frame(v3,p3,row.names=c("Out","3rd","Home"))
      ts <- thirdsteal[sample(nrow(thirdsteal), 1, replace = TRUE, prob = thirdsteal$p3), -ncol(thi
      if (ts == 0){
        og_score <- c(0,0,0,og_score[4]+1,og_score[5])
        trial0 <- nobodyon(batterprobs[,i])
        score <- c(og_score[1:3] != trial0[1:3], og_score[4:5] + trial0[4:5])
    }
      if (ts == 2){
        og_score <- c(0,0,0,og_score[4],og_score[5]+1)
        trial0 <- nobodyon(batterprobs[,i])
        score <- c(og_score[1:3] != trial0[1:3], og_score[4:5] + trial0[4:5])
    }
      if (ts == 1){
        trial3 <- manonthird(batterprobs[,i])
        score <- c(og_score[1] != trial3[1], og_score[2] != trial3[2], og_score[3] == trial3[3], og
  }}
      if (og_score[1] == 1 & og_score[2] == 1 & og_score[3] == 0){
        v4 <- c(0,1,2)
        p4 <- c((s[1,4])*(1-(s[2,4])),1-s[1,4],(s[1,4]*s[2,4]))
        firstandsecondsteal <- data.frame(v4,p4,row.names=c("Out","Safe","Xtra"))
        fss <- firstandsecondsteal[sample(nrow(firstandsecondsteal), 1, replace = TRUE, prob = firs
        if (fss == 0){
          og_score <- c(0,1,0,og_score[4]+1,og_score[5])
          trial2 <- manonsecond(batterprobs[,i])
          score <- c(og_score[1] != trial2[1], og_score[2] == trial2[2], og_score[3] != trial2[3], 
    }
        if (fss == 2){
          og_score <- c(0,1,1,og_score[4],og_score[5])
    }
        if (fss == 1){
          trial4 <- manonfirstandsecond(batterprobs[,i])
          score <- c(og_score[1] == trial4[1], og_score[2] == trial4[2], og_score[3] != trial4[3], 
  }}
        if (og_score[1] == 0 & og_score[2] == 1 & og_score[3] == 1){
          v3 <- c(0,1,2)
          p3 <- c((s[1,3])*(1-(s[2,3])),1-s[1,3],(s[1,3]*s[2,3]))
          thirdsteal <- data.frame(v3,p3,row.names=c("Out","3rd","Home"))
          ts <- thirdsteal[sample(nrow(thirdsteal), 1, replace = TRUE, prob = thirdsteal$p3), -ncol
          if (ts == 0){
```

```
                og_score <- c(0,0,1,og_score[4]+1,og_score[5])
                trial3 <- manonthird(batterprobs[,i])
                score <- c(og_score[1] != trial3[1], og_score[2] != trial3[2], og_score[3] == trial3[3]
          }
              if (ts == 2){
                og_score <- c(0,0,1,og_score[4],og_score[5]+1)
                trial3 <- manonthird(batterprobs[,i])
                score <- c(og_score[1] != trial3[1], og_score[2] != trial3[2], og_score[3] == trial3[3]
        }
              if (ts == 1){
                trial5 <- manonsecondandthird(batterprobs[,i])
                score <- c(og_score[1] != trial5[1], og_score[2] == trial5[2], og_score[3] == trial5[3]
      }}
              if (og_score[1] == 1 & og_score[2] == 1 & og_score[3] == 1){
                  trial6 <- basesloaded(batterprobs[,i])
                  score <- c(og_score[1:3] == trial6[1:3], og_score[4:5] + trial6[4:5])
      }
              if (og_score[1] == 1 & og_score[2] == 0 & og_score[3] == 1){
                v5 <- c(0,1,2)
                p5 <- c((s[1,5])*(1-(s[2,5])),1-s[1,5],(s[1,5]*s[2,5]))
                firstandthirdsteal <- data.frame(v5,p5,row.names=c("Out","1st","2nd"))
                fts <- firstandthirdsteal[sample(nrow(firstandthirdsteal), 1, replace = TRUE, prob = fi
                if (fts == 0){
                  og_score <- c(0,0,1,og_score[4]+1,og_score[5])
                  trial3 <- manonthird(batterprobs[,i])
                  score <- c(og_score[1] != trial3[1], og_score[2] != trial3[2], og_score[3] == trial3[
        }
                if (fts == 2){
                  og_score <- c(0,1,1,og_score[4],og_score[5])
                  trial5 <- manonsecondandthird(batterprobs[,i])
                  score <- c(og_score[1] != trial5[1], og_score[2] == trial5[2], og_score[3] == trial5[
        }
                if (fts == 1){
                  trial7 <- manonfirstandthird(batterprobs[,i])
                  score <- c(og_score[1] == trial7[1], og_score[2] != trial7[2], og_score[3] == trial7[
      }}
    score}
```

```
runsbatters <- c(0,0)
```

```
inningruns <- function(outs,i,s){
  totalruns <- c(0)
  score <- c(0,0,0,0,0)
  while (length(score) == 5){
      a <- singlebatter(score,i,s)
      if (i == 9){
        i <- 0
      }
      a <- singlebatter(a,i+1,s)
      if (i == 8){
        i <- -1
      }
      a <- singlebatter(a,i+2,s)
```

```
    if (a[4] >= 3){
      runsbatters <- c(a[5],3)
      break
    }
    if (i == 7){
      i <- -2
    }
    a <- singlebatter(a,i+3,s)
    if (a[4] >= 3){
      runsbatters <- c(a[5],4)
      break
    }
    if (i == 6){
      i <- -3
    }
    a <- singlebatter(a,i+4,s)
    if (a[4] >= 3){
      runsbatters <- c(a[5],5)
      break
    }
    if (i == 5){
      i <- -4
    }
    a <- singlebatter(a,i+5,s)
    if (a[4] >= 3){
      runsbatters <- c(a[5],6)
      break
    }
    if (i == 4){
      i <- -5
    }
    a <- singlebatter(a,i+6,s)
    if (a[4] >= 3){
      runsbatters <- c(a[5],7)
      break
    }
    if (i == 3){
      i <- -6
    }
    a <- singlebatter(a,i+7,s)
    if (a[4] >= 3){
      runsbatters <- c(a[5],8)
      break
    }
    if (i == 2){
      i <- -7
    }
    a <- singlebatter(a,i+8,s)
    if (a[4] >= 3){
      runsbatters <- c(a[5],0)
      break
    }
    if (i == 1){
```

```
  i <- -8
}
a <- singlebatter(a,i+9,s)
if (a[4] >= 3){
  runsbatters <- c(a[5],1)
  break
}
if (i == -1){
  i <- 8
}
if (i == -2){
  i <- 7
}
if (i == -3){
  i <- 6
}
if (i == -4){
  i <- 5
}
if (i == -5){
  i <- 4
}
if (i == -6){
  i <- 3
}
if (i == -7){
  i <- 2
}
if (i == -8){
  i <- 1
}
a <- singlebatter(a,i+1,s)
if (a[4] >= 3){
  runsbatters <- c(a[5],2)
  break
}
if (i == 8){
  i <- -1
}
a <- singlebatter(a,i+2,s)
if (a[4] >= 3){
  runsbatters <- c(a[5],3)
  break
}
if (i == 7){
  i <- -2
}
a <- singlebatter(a,i+3,s)
if (a[4] >= 3){
  runsbatters <- c(a[5],4)
  break
}
if (i == 6){
```

```
        i <- -3
      }
      a <- singlebatter(a,i+4,s)
      if (a[4] >= 3){
        runsbatters <- c(a[5],5)
        break
      }
      if (i == 5){
        i <- -4
      }
      a <- singlebatter(a,i+5,s)
      if (a[4] >= 3){
        runsbatters <- c(a[5],6)
        break
      }
      if (i == 4){
        i <- -5
      }
      a <- singlebatter(a,i+6,s)
      if (a[4] >= 3){
        runsbatters <- c(a[5],7)
        break
      }
      if (i == 3){
        i <- -6
      }
      a <- singlebatter(a,i+7,s)
      if (a[4] >= 3){
        runsbatters <- c(a[5],8)
        break
      }
      if (i == 2){
        i <- -7
      }
      a <- singlebatter(a,i+8,s)
      if (a[4] >= 3){
        runsbatters <- c(a[5],0)
        break
      }
      else{
        runsbatters <- c(a[5],1)
        break
      }
  }
  runsbatters
}
```

```
game <- function(i,s){
  runsbatters <- c(0,0)
  score <- c(0,0,0,0,0)
  outs <- c(0)
  i1 <- inningruns(0,i,s)
  i2 <- inningruns(0,i1[2]+1,s)
```

```
  i3 <- inningruns(0,i2[2]+1,s)
  i4 <- inningruns(0,i3[2]+1,s)
  i5 <- inningruns(0,i4[2]+1,s)
  i6 <- inningruns(0,i5[2]+1,s)
  i7 <- inningruns(0,i6[2]+1,s)
  i8 <- inningruns(0,i7[2]+1,s)
  i9 <- inningruns(0,i8[2]+1,s)
  c(sum(i1[1],i2[1],i3[1],i4[1],i5[1],i6[1],i7[1],i8[1],i9[1]))
}
```

```
a <- replicate(100,game(1,stealmatrix(0.5,0.9)))
b <- replicate(100,game(1,stealmatrix(0,0)))
```

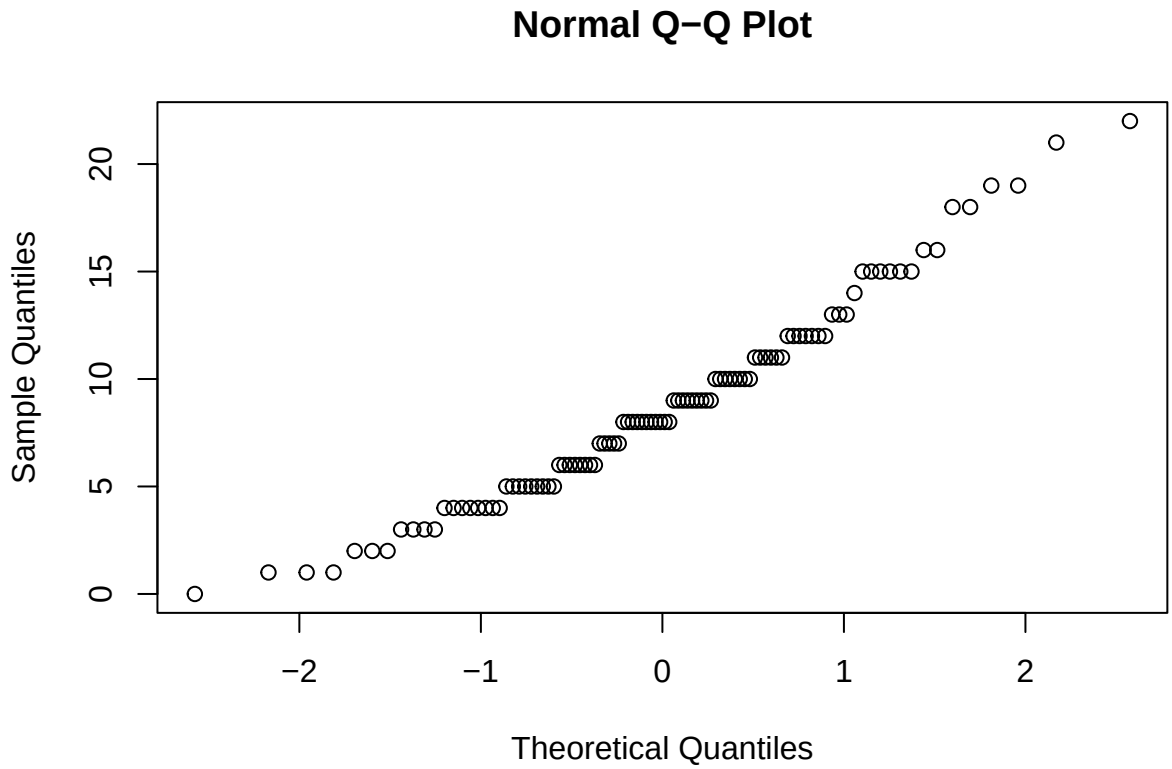
```
mean(a)
```

```
## [1] 8.72
```

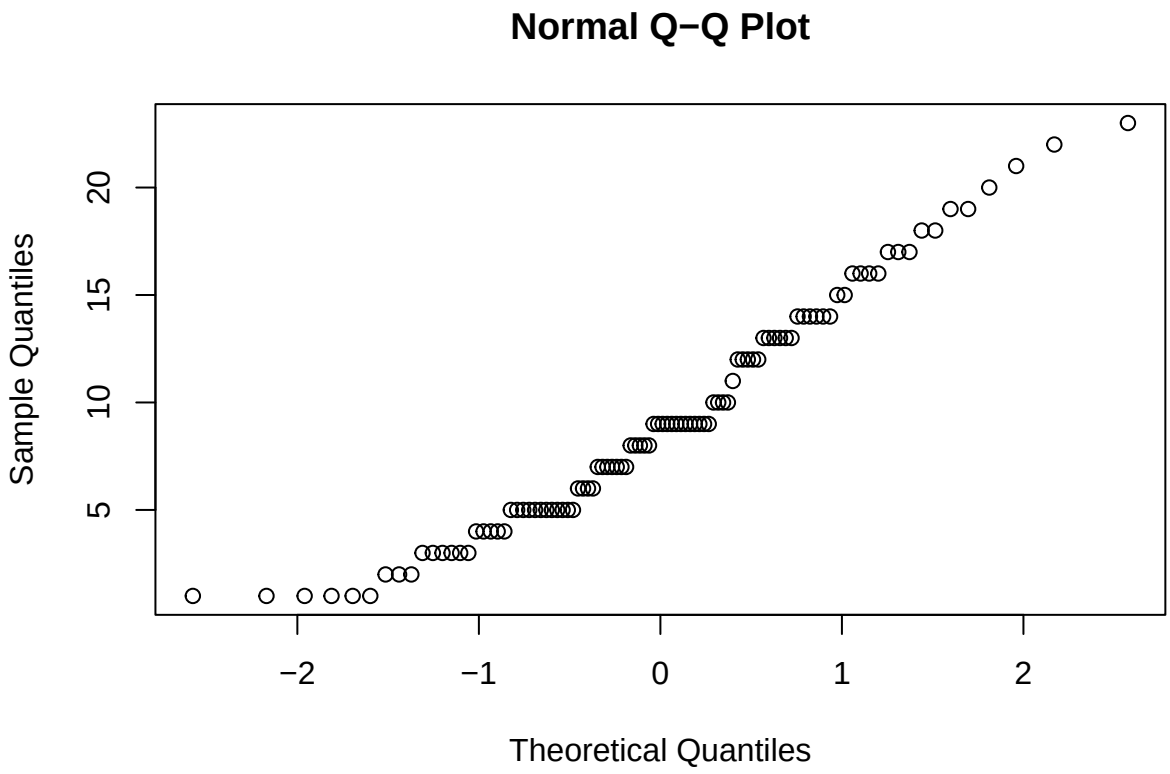
```
mean(b)
```

```
## [1] 9.18
```

```
qqnorm(a)
```



```r
qqnorm(b)
```




```r
set.seed(100432748)
stealrates1 <- seq(0,0.85,by=0.01)
runspergame1 <- replicate(length(stealrates1),c(1))
for (i in 1:length(stealrates1)){
  runspergame1[i] <- replicate(1000,game(1,stealmatrix(stealrates1[i],su))) %>% mean()
}
#plot(stealrates,runspergame)
```

```r
stealrates <- seq(0,0.85,by=0.01)
runspergame <- replicate(length(stealrates),c(1))
for (i in 1:length(stealrates)){
  runspergame[i] <- replicate(50,game(1,stealmatrix(stealrates[i],su))) %>% mean()
}
```

```r
data <- data.frame(stealrates1,runspergame1)
reg <- lm(runspergame1~stealrates1+I(stealrates1^2),data)
new.data <- data.frame(steal = seq(from = min(data$stealrates1),
                                   to = max(data$stealrates1), length.out = 86))
pred1 <- predict(reg, newdata = new.data)
stealrates1[42]
```

```
## [1] 0.41
```

```r
pred1[42]
```

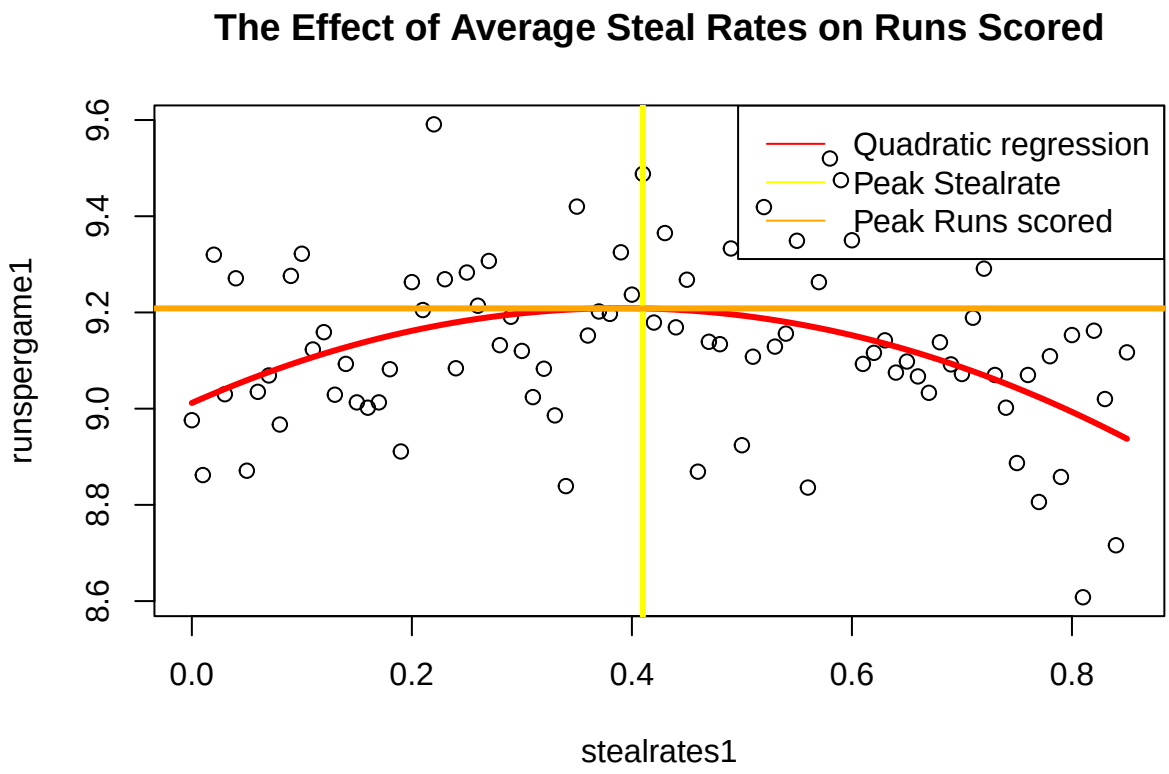
```
##       42
## 9.208412
```

```r
summary(reg)
```

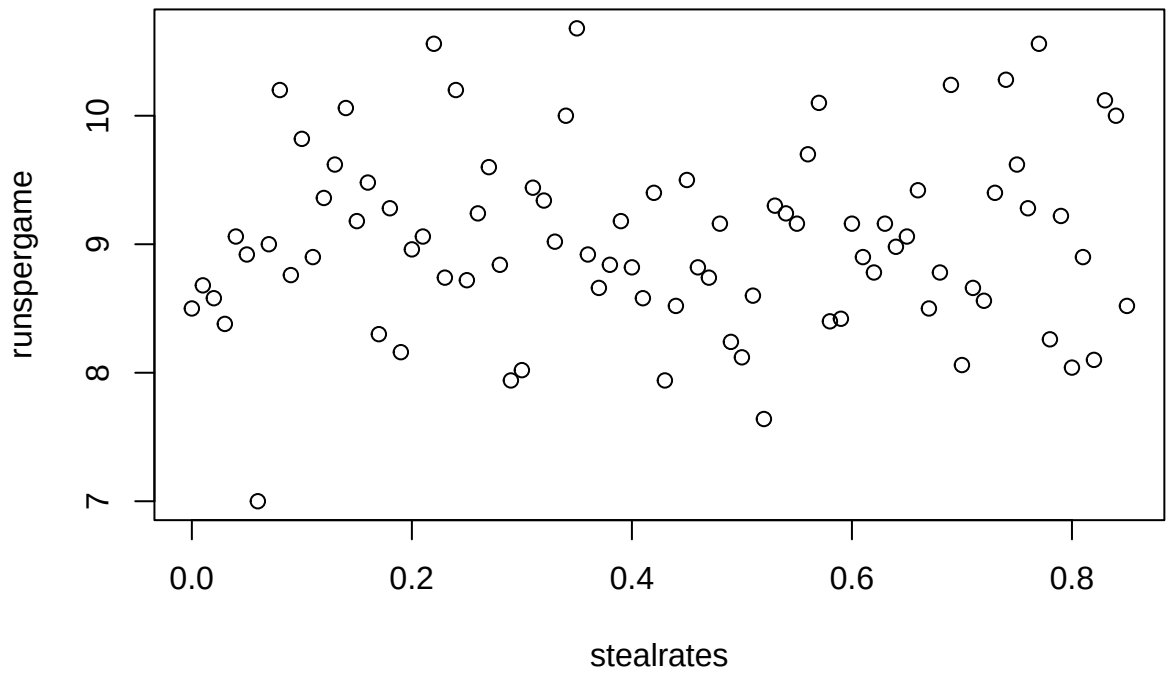
```
##
## Call:
## lm(formula = runspergame1 ~ stealrates1 + I(stealrates1^2), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.37461 -0.08152 -0.01277  0.10079  0.41981
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       9.01185    0.05183 173.887  < 2e-16 ***
## stealrates1       1.00783    0.28183   3.576 0.000585 ***
## I(stealrates1^2) -1.28879    0.32080  -4.017 0.000129 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1639 on 83 degrees of freedom
## Multiple R-squared:  0.1754, Adjusted R-squared:  0.1555
## F-statistic: 8.827 on 2 and 83 DF,  p-value: 0.0003343
```

The Quadratic Regression Line is very significant with the 1000 game replications for each steal rate. The line equation: Runs Scored = 7.81561 + 0.96403(Stealrate) - 1.18352(Stealrate)^2

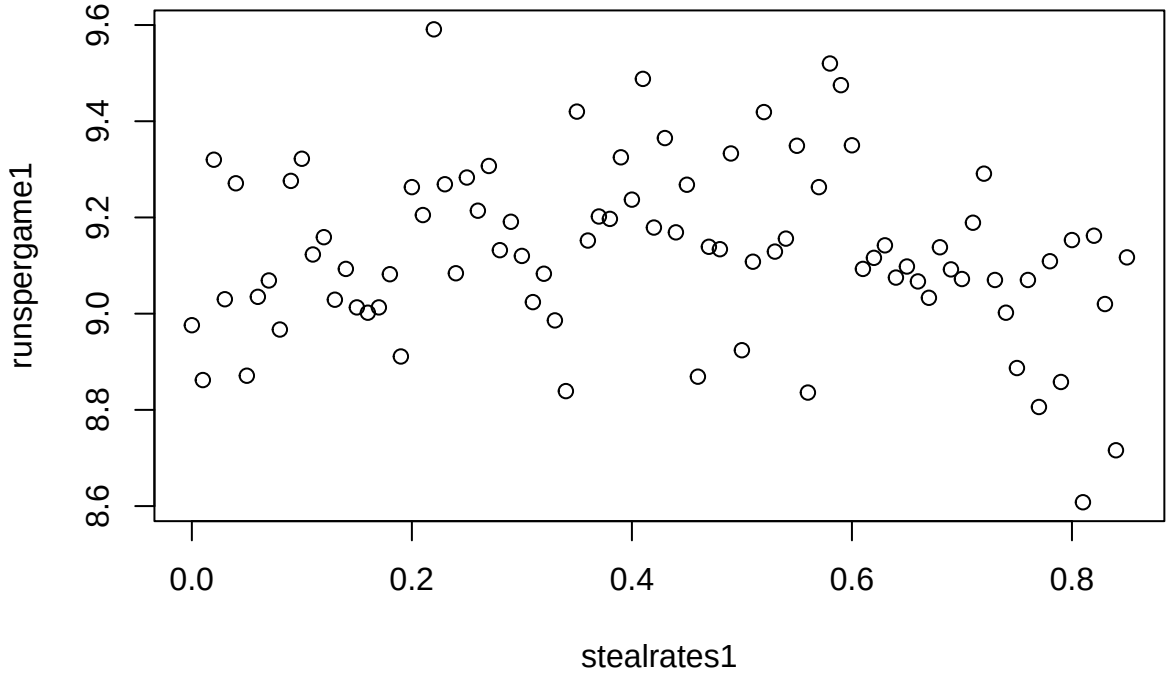
```r
a1000 <- plot(stealrates1,runspergame1,main="The Effect of Average Steal Rates on Runs Scored")
lines(pred1 ~ new.data$steal, col = "red",lwd=3)
abline(v=stealrates1[42],col="yellow",lwd=3)
abline(h=pred1[42],col="orange",lwd=3)
legend("topright", c("Quadratic regression","Peak Stealrate","Peak Runs scored"),col=c("red","yellow","
```




```
a50 <- plot(stealrates,runspergame)
```




```
a1000 <- plot(stealrates1,runspergame1)
```



The Quadratic Regression Line is very significant with the 1000 game replications for each steal rate. The line equation: Runs Scored = 7.81561 + 0.96403(Stealrate) - 1.18352(Stealrate)ˆ2