Домашно 1

#Домашно номер 1 ###Николай Кормушев 81805.

```
library("ggplot2")
library("tidyverse")
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
## v tibble  3.1.0     v dplyr   1.0.4
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
## v purrr   0.3.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
colnames(diamonds)
```

```
##  [1] "carat"   "cut"     "color"   "clarity" "depth"   "table"   "price"
##  [8] "x"       "y"       "z"
```

###Задача 1 #####а

```
nrow(diamonds[diamonds$price > 15000,])
```

```
## [1] 1655
```

#####б

```
nrow(diamonds[diamonds$carat < 2 & diamonds$price > 15000,])
```

```
## [1] 498
```

#####в

```
m <- mean(diamonds$price)
s <- sd(diamonds$price)
nrow(diamonds[diamonds$price < m + s & diamonds$price > m - s,])
```

```
## [1] 46225
```

#####г

```
ideal <- diamonds[diamonds$cut == "Ideal",]
quantile(ideal$carat)
```

```
##   0%  25%  50%  75% 100%
## 0.20 0.35 0.54 1.01 3.50
```

```
median(ideal$carat)
```

```
## [1] 0.54
```

```
mean(ideal$carat)
```

```
## [1] 0.702837
```

#####д Цена, такава че 80% от диамантите са под нея

```
quantile(diamonds$price, prob = 0.8)
```

```
##    80%
## 6301.2
```

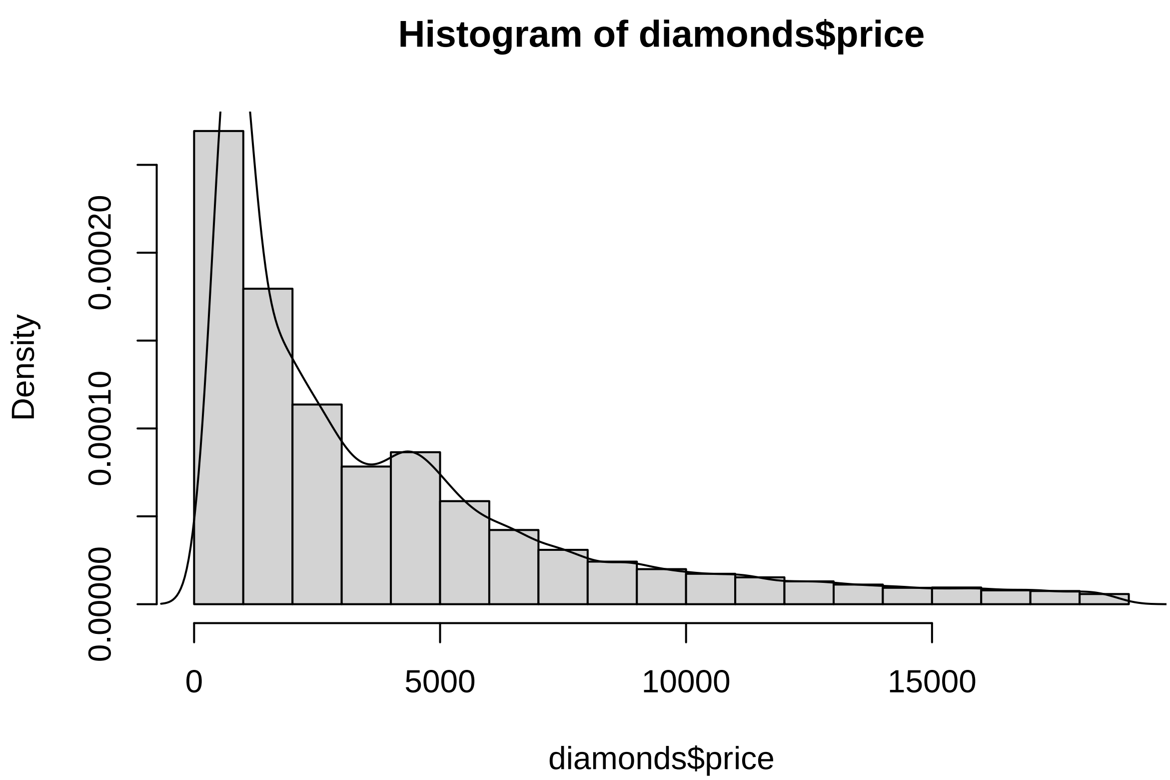
#####е Диаманите сортирани по x, y, z

```
diamonds[order(diamonds$x, diamonds$y, diamonds$z, decreasing = T),][1:5,]
```

```
## # A tibble: 5 x 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  5.01 Fair      J     I1       65.5    59 18018  10.7 10.5   6.98
## 2  4.5  Fair      J     I1       65.8    58 18531  10.2 10.2   6.72
## 3  4.01 Premium   I     I1       61      61 15223  10.1 10.1   6.17
## 4  4.01 Premium   J     I1       62.5    62 15223  10.0  9.94  6.24
## 5  4    Very Good I     I1       63.3    58 15984  10.0  9.94  6.31
```

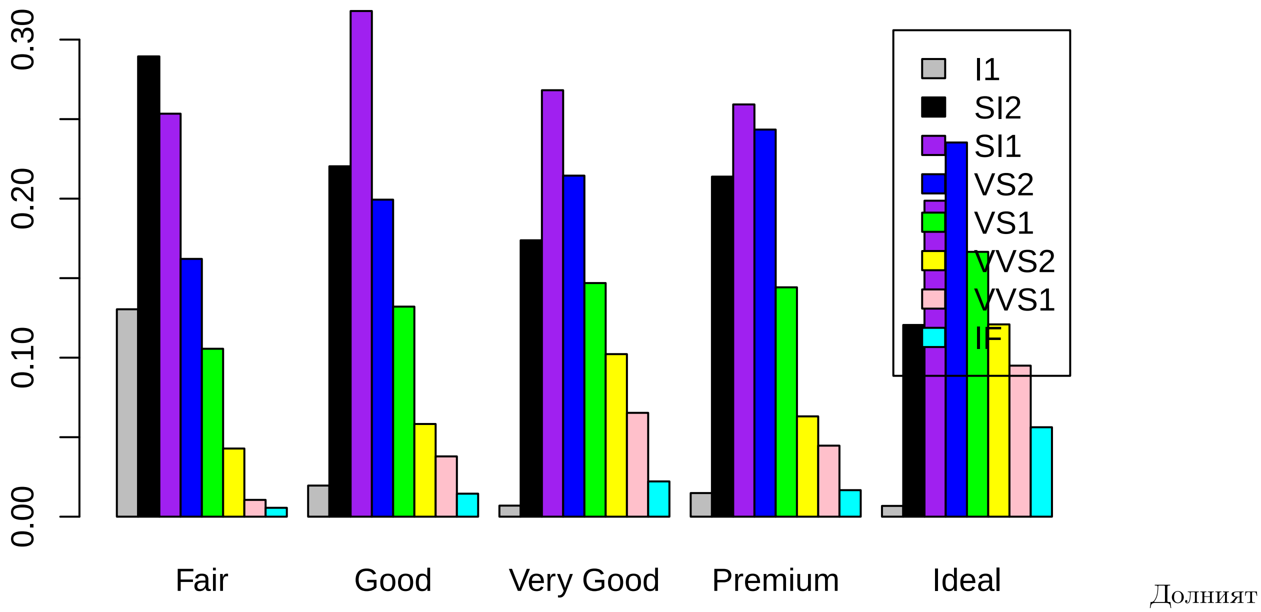
###Задача 2 #####а

```
hist(diamonds$price, probability = T)
lines(density(diamonds$price))
```



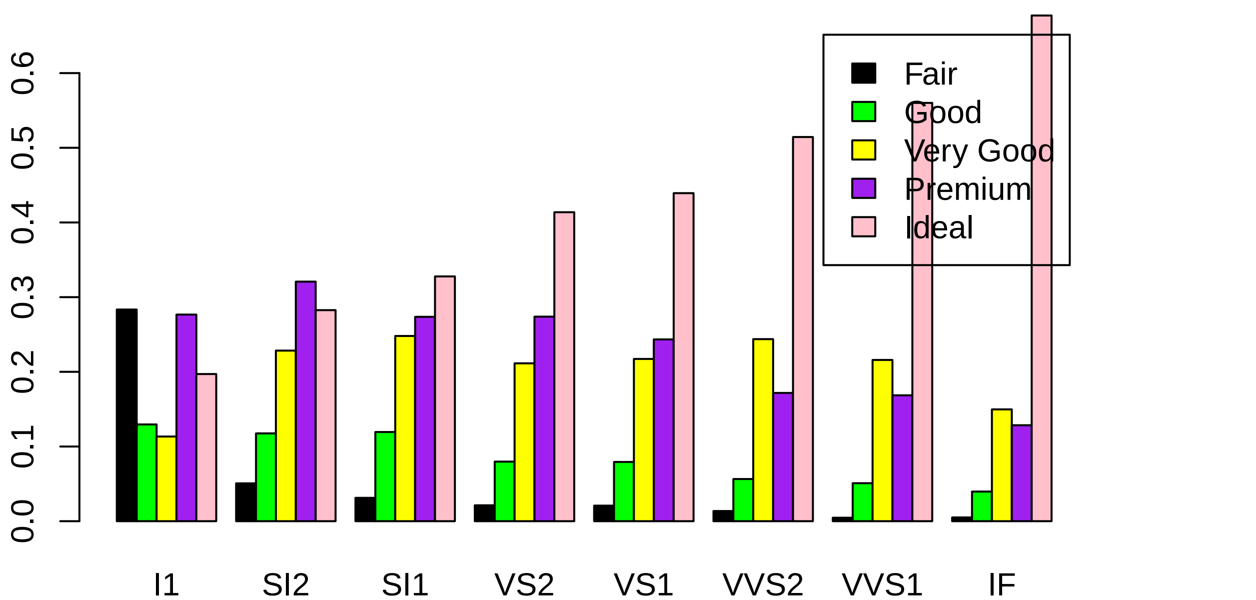
#####б Долният barplot показва условнаta вероятност, ако знаем каква е шлифовката каква е вероятността да имаме диамант с дадена чистота. Чистотата е подредена по качество. Тоест най-долу на легендата е най-хубава(IF), а най-горе е най-лоша(I1). На диаграмата се вижда, че колкото по-добра е шлифовката толкова по-висока е вероятността диамантът да е с по-хубава чистота. Тоест ако шлифовката е идеално е доста по-вероятно да имаме супер чист(IF) диамант отколкото, ако е ставаща(Fair)

```
barplot(prop.table(table(diamonds$clarity, diamonds$cut), 2), legend = T,
        beside = T, col = c("grey", "black", "purple", "blue", "green",
        "yellow", "pink", "cyan"))
```



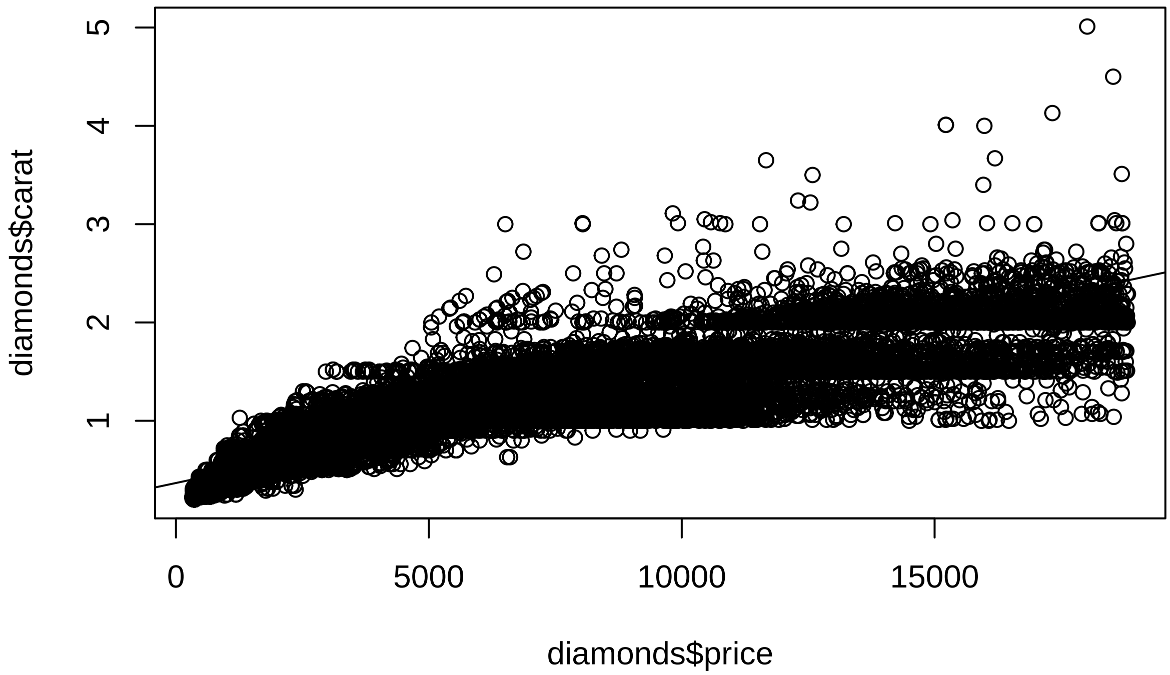
Долният barplot показва каква е вероятността, ако знаем чистотата да имаме някоя от шлифовките. Както се вижда колкото по-чист ни е диамантът толкова по-вероятно е да е с по-хубава шлифовка. Колкото по-мръсен е вероятностите за качествена и некачествена шлифовка са доста по-близки. Вижда се все пак, че дори от най-замърсените диаманти можеш да получиш идеална шлифовка, което също е интересно.

```
barplot(prop.table(table(diamonds$cut, diamonds$clarity), 2), legend = T,
        beside = T, col = c( "black", "green", "yellow", "purple", "pink"))
```



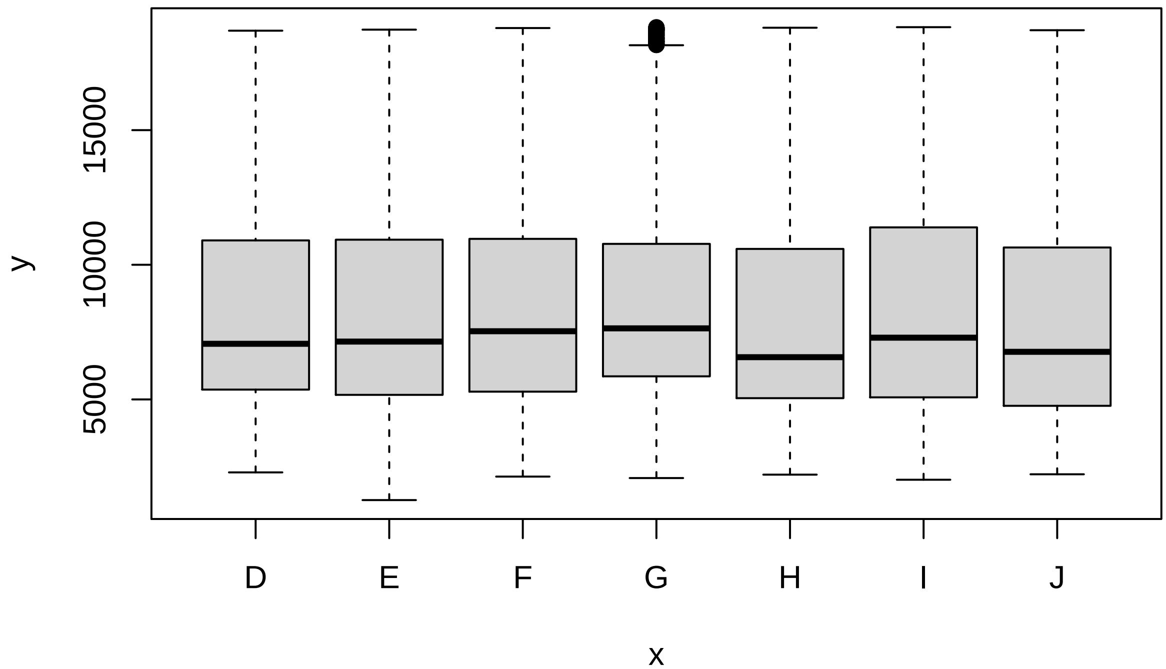
#####в Долната графика покзва каратите на диамантите и съответната им цена разпределени в 2d равнина. От нея се вижда, че са зависи, както и може да се начертае проста линейна регресия, която да описва връзката между данните.

```
plot(diamonds$price, diamonds$carat)
l = lm(carat ~ price, data = diamonds)
abline(l)
```

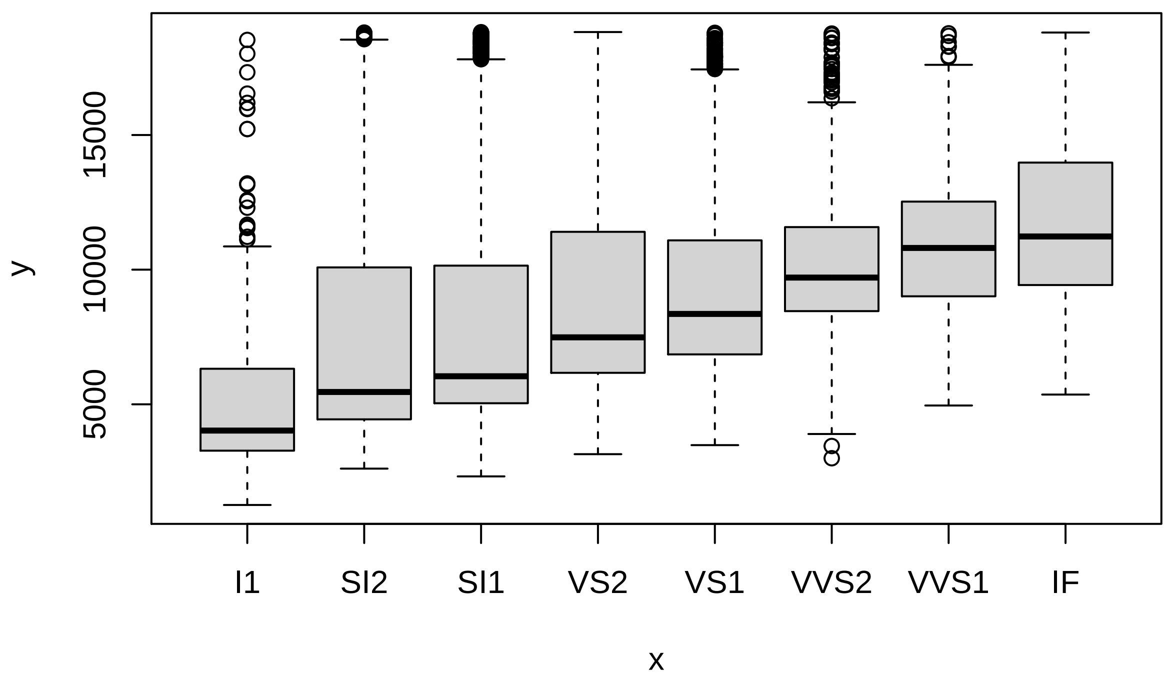


#####г От трите диаграми се вижда, че чистотата влияе най-силно на цената. Най-скъпи са чистите диаманти. Интересно е, че ако включа и диамантите под 1 карат диаграмите изглеждаха доста различно и беше много странно, защото тогава цветът, като че ли играеше основна роля. Колко е чист диаманта почти не беше от значение. Даже най-чистите бяха доста нодолу според boxplot-a(разбира се с много outliers)

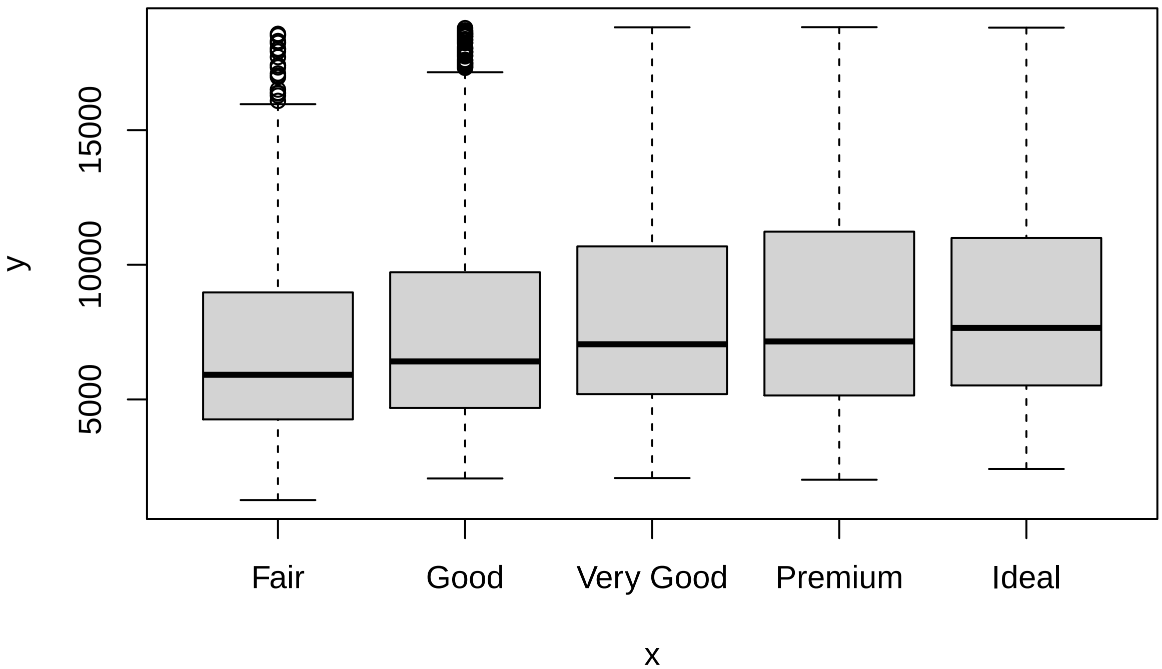
```
caratsMoreThan1 <- diamonds[diamonds$carat > 1,]

plot(caratsMoreThan1$color, caratsMoreThan1$price)
```




```
plot(caratsMoreThan1$clarity, caratsMoreThan1$price)
```




```
plot(caratsMoreThan1$cut, caratsMoreThan1$price)
```



###Задача 3 #####a

i не се ползва. По-долу обяснявам защо. Реших да си поиграя с функции от по-висок ред общо взето

```
add1 <- function (x, i) {
  x <- x + 1
}
```

Изпълнение на един опит. По подразбиране следва точно описания в условието алгоритъм:

```
attempts = 10000

draw <- function(whiteNum = 2, blackBalls = 2, ballNum = 4, N = 20, fn = add1, result = 0) {
  for (i in 1:N) {
    ball <- sample(c("white", "black"), 1,
                   prob = c(whiteNum/ballNum, blackBalls/ballNum))
    if(ball == "white") {
      whiteNum <- whiteNum + 2
      ballNum  <- ballNum + 2
      result <- fn(result, i)
    } else {
      blackBalls <- blackBalls + 1
      ballNum  <- ballNum + 1
    }
  }

  result
}
```

Резултати от attempts брой опита. Записваме на всеки опит колко бели топки сме извадили в списък

```
whiteBalls <- 1:attempts %>%
  map(~draw()) %>%
  unlist()
```

Функция смятаща емпиричната вероятност.

```
eProb <- function(n, whiteBalls, attempts) {
  sum(whiteBalls == n)/attempts
}
```

Емпиричната вероятност X = 15

```
eProb(15, whiteBalls, attempts)
```

```
## [1] 0.0782
```

Емпиричната вероятност X < 10

```
sum(whiteBalls < 10)/attempts
```

```
## [1] 0.2388
```

#####б Смятаме разпределението на X и после медианата и квантилите

```
xDistribution <- 0:20 %>%
  map(~eProb(.x, whiteBalls, attempts)) %>%
  unlist()

median(xDistribution)
```

```
## [1] 0.0479
```

```
quantile(xDistribution)
```

```
##     0%    25%    50%    75%   100%
## 0.0112 0.0250 0.0479 0.0730 0.0890
```

#####в Индексът съвпада с броя извадени бели топки Взимаме на кой индекс е максималната вероятност И получаваме най-вероятният брой бели топки Отдолу добавих и самата вероятност

```
which.max(xDistribution)
```

```
## [1] 17
```

```
max(xDistribution)
```

```
## [1] 0.089
```

#####г Теглили сме 2 черни топки и 0 бели. Значи сме с +2 черни топки. Тоест имаме 4 черни и 2 бели и общо 6 топки. Ще симулирам само последните тегления. Започваме от 2

Ползвам функции от по-висок ред,да намаля повтарянето на код. Затова fifthBall и add1 приемат бонус ненужен аргумент, което не е много добре от гледна точка качество на код, ама върши работа

Тук гледаме дали на третото теглене се вика тази функция. Ако се вика, то петата изтеглена топка е бяла, защото в този случай се вика функцията i == 3, защото вече сме теглили два пъти. Общо взето слага стойността на result на TRUE, ако при последното теглене получим бяла.

```
fifthBall <- function (x, i) {
  i == 3
}
```

Резултатите от опитите ги слагам в списък

```
fifthIsWhite <- 1:attempts %>%
  map(~draw(2, 4, 6, 3, fifthBall, FALSE)) %>%
  unlist()
```

Виждам броя пъти, в които петата топка е била бяла и деля на броя опити и получавам емпиричната вероятност.

```
sum(fifthIsWhite)/attempts
```

```
## [1] 0.3835
```

#####д Общо взето умножавам на всеки ход по вероятността да извадим бяла топка и добавям двете нови бели топки. Вероятността за даден ход е броя бели топки на този ход делено на общия брой топки за хода. Най-отдолу се вижда, че теоритичната и получената емпирична вероятност са доста близки.

```
tProb <- 1
whiteNum <- 2
ballNum <- 4
for (i in 1:20) {
  tProb <- tProb * whiteNum/ballNum

  whiteNum <- whiteNum + 2
  ballNum <- ballNum + 2
}

tProb
```

```
## [1] 0.04761905
```

```
eProb(20, whiteBalls, attempts)
```

```
## [1] 0.0523
```