

Class Work 1

Justas Mundeikis

2019-03-14

1 Nesažiningų kauliukų Casino

1.1 Įvadas

Šio užsiėmimo tikslai:

- susipažinti su kai kuriomis naujomis funkcijomis
- įprasti rašyti nuosavas funkcijas Šis užsiėmimas remiasi “Hands-On Programming with R” (Garett Grolemund) Šio užsiėmimo metu sukonstruosime funkciją, kuri imituos sžiningo kauliuko metimą, kai galėsime mesti kauliukus ir gauti atsitiktinius skaičius. Tačiau norėdami užsidirbti daug pinigų, šiek tiek tweekinsim kauliukus, jog jie būtų labiau mūsų, t.y. Casino, naudai.

1.2 Prisiminimui

Sukurkite obejktą “kauliukas”. Priminimas: objektų pavadinimai negali prasidėti skaičiumi, \$, ^ bei kitais aritmetiniais simboliais

```
kauliukas <- 1:6
kauliukas
## [1] 1 2 3 4 5 6
```

Siekiant sužinoti, kokius objektus jau esate sukūrę ir kurie yra R darbo atmintyje, galite naudooti `ls()` komandą (panašiaip kaip ir CLI). Jeigu norite žinoti, kokie failai yra Workspace `dir()`

```
ls()
## [1] "kauliukas"
# getwd()
# dir()
```

R atlieka vektorizuotas operacijas, todėl galime naudoti panašias komandas:

```
kauliukas - 1
## [1] 0 1 2 3 4 5
kauliukas * 2
## [1] 2 4 6 8 10 12
kauliukas / 3
## [1] 0.3333333 0.6666667 1.0000000 1.3333333 1.6666667 2.0000000
kauliukas * kauliukas
## [1] 1 4 9 16 25 36
```

Tačiau atsargiai su *vector recycling*, jeigu vienas vektorius yra trumpesnis už kitą, R jį perkramtys ir panaudos tiek, kad atliktų visas reikalingas matematines operacijas.

```
kauliukas + c(1,2,3)
## [1] 2 4 6 5 7 9
kauliukas + c(1,2,3,4)
## Warning in kauliukas + c(1, 2, 3, 4): longer object length is not a
## multiple of shorter object length
## [1] 2 4 6 8 6 8
```

1.3 Funkcijos

Funkcijos priima argumentus

```

mean(1:6)
## [1] 3.5
mean(kauliukas)
## [1] 3.5
round(pi)
## [1] 3
round(mean(kauliukas))
## [1] 4

```

Norint galėti mesti kauliukus, galima naudotis funkcija `sample`

```

sample(x=1:4, size = 2)
## [1] 2 4
sample(kauliukas, size = 1)
## [1] 1
sample(kauliukas, size = 1)
## [1] 2
sample(kauliukas, size = 1)
## [1] 4
sample(kauliukas, size = 1)
## [1] 2

```

Jeigu pamirštate, kokius argumentus priima funkcija, galite naudotis `args()` funkcija

```

args(round) #čia jau predefinuotas, jog apvalinimas vykty iki 0 ženklų po kablelio
## function (x, digits = 0)
## NULL
round(pi, digits=1)
## [1] 3.1
round(pi, digits=2)
## [1] 3.14
round(pi, digits=3)
## [1] 3.142
args(sample)
## function (x, size, replace = FALSE, prob = NULL)
## NULL
sample(size=1, kauliukas)
## [1] 3
sample(kauliukas, size=6)
## [1] 2 5 4 6 1 3
sample(kauliukas, size = 6, replace = TRUE)
## [1] 4 5 1 6 4 3

```

Tarkime “kauliukai” tai išmestų dviejų kauliukų suma

```

kauliukai <- sample(kauliukas, size=2, replace = TRUE)
kauliukai
## [1] 2 6
sum(kauliukai)
## [1] 8
# jeigu dabar keliskart iššauksime kauliukai:
kauliukai
## [1] 2 6
kauliukai
## [1] 2 6
kauliukai
## [1] 2 6
#kaskart gausime jau išsaugotas kauliukai reikšmės

```

1.3.1 Uždavinys

1.3.1.1 Užduotis

Parašykite funkciją `metimas()`, kuri išmestų ir susumuočių išmestų 2 kauliukų sumą. Priminimui:

```

kauliukas <- 1:6
kauliukai <- sample(kauliukas, size=2, replace = TRUE)
sum(kauliukai)

```

1.3.1.2 Sprendimas

```

metimas <- function(){
  kauliukas <- 1:6
  kauliukai <- sample(kauliukas, size=2, replace = TRUE)
  sum(kauliukai) #svarbu iššaukti rezultatą
}

metimas()
## [1] 7
metimas()
## [1] 4
metimas()
## [1] 6

```

1.3.2 Ir dar

Jeigu pakeisime funkciją iš `sample(kauliukas...)` į `sample(kaulas...)`

```

metimas2 <- function(){
  kauliukai <- sample(kaulas, size=2, replace = TRUE)
  sum(kauliukai)
}

metimas2()
## Error in sample(kaulas, size = 2, replace = TRUE): object 'kaulas' not found

```

Bet galime perrašyti funkciją taip

```

metimas2 <- function(kaulas){
  kauliukai <- sample(kaulas, size=2, replace = TRUE)
  sum(kauliukai)
}
# funkcija neveiks nenurodant kaulas argumento
metimas2()
## Error in sample(kaulas, size = 2, replace = TRUE): argument "kaulas" is missing, with no default
# dabar funkcija veiks, jeigu argumentui bus priskirtos reikšmės
metimas2(kaulas=1:4)
## [1] 5
metimas2(kaulas=1:10)
## [1] 10
metimas2(kaulas=100:200)
## [1] 244

```

Jeigu norime, galime predefinuoti, kas yra “kaulas” t.y. argumentui kaulas priskirti iš anksto numatytas reikšmės

```

metimas2 <- function(kaulas=1:6){
  kauliukai <- sample(kaulas, size=2, replace = TRUE)
  sum(kauliukai) #svarbu iššaukti rezultatą
}
metimas2()
## [1] 7

```

1.4 Grafikai

Sugeneruojame du vektorius:

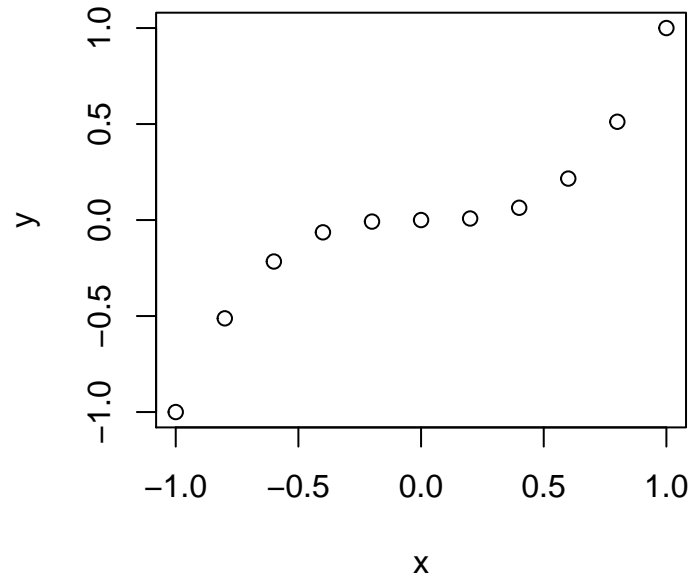
- x ašį nuo -1 iki 1

- ir funkciją $y=x^3$

```
x <- seq(from=-1, to=1, by=0.2)
y <- x^3
```

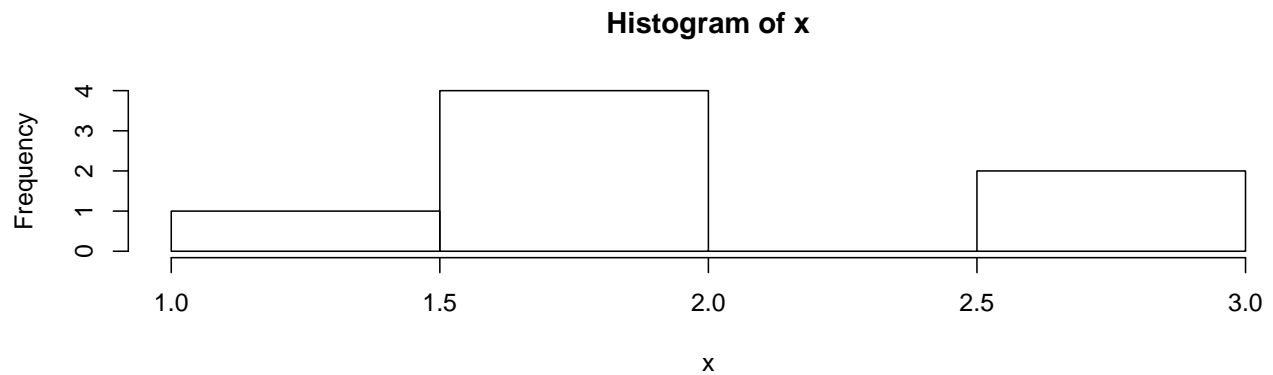
Dabar galime nubraižyti jų grafiką

```
plot(x,y)
```

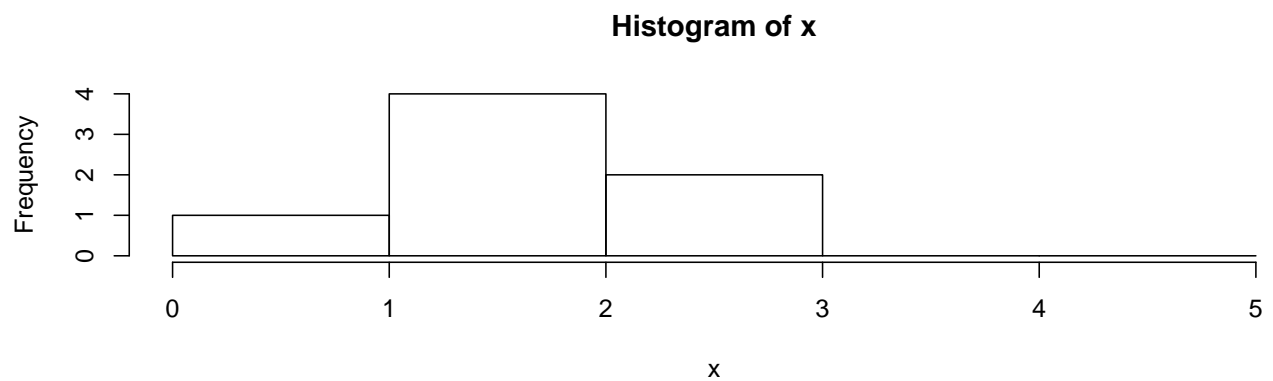


Galime sugeneruoti ir histogramą:

```
x <- c(1,2,2,2,2,3,3)
hist(x)
```



```
hist(x, breaks = c(0:5))
```



1.4.1 Uždavinys

1.4.1.1 Užduotis

funkcija `rnorm()` generuoja normaliojo skirstinio skaičius. Jeigu reikia pagalbos, `?rnorm`. Nubraižykite:

- 100 verčių iš normaliojo skirstinio,
- kurio vidurkis =100,
- o standartinis nuokrypis 1 histogramą.

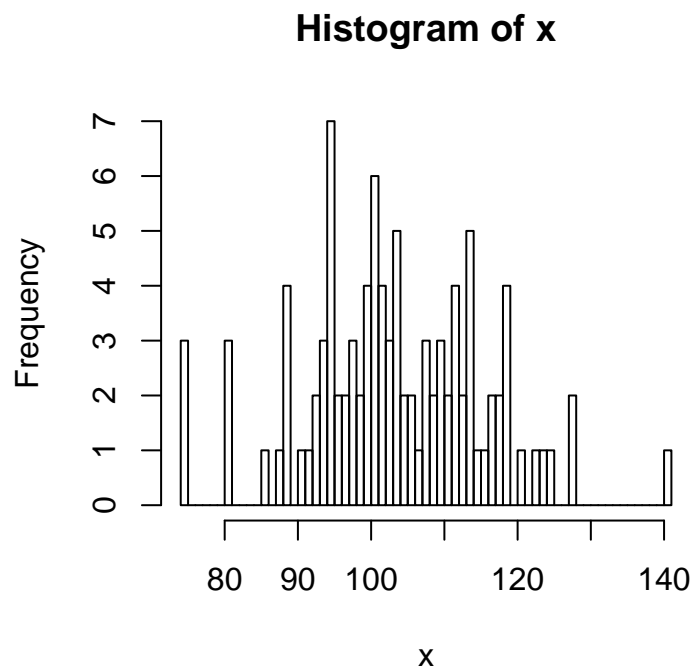
Histogramos x ašies vertės turėtų būti:

- nuo mažiausios x vertės suapvalinus žemyn
- iki didžiausios x vertės suapvalinus į viršų
- skirtumas 1

Pagalba dėl apvalinimo `?round`

1.4.1.2 Sprendimas

```
x <- rnorm(n = 100, mean = 100, sd = 10)
hist(x,
      breaks = seq(from=floor(min(x)),
                    to=ceiling(max(x)),
                    by=1)
)
```



1.4.2 Funkcijos replikavimas (=sapply)

Grįžkime prie mūsų lošimo.

Komanda `replicate()` leidžia kartoti tam tikrą R komandą norimą kartų kiekį, arba pvz pakartoti tam tikrą vektoriu. Pastaba `replicate(n, expr,...) = sapply(1:n, function(x) call)`

```
args(replicate)
## function (n, expr, simplify = "array")
## NULL
# puz pakartotja 3 kartus komanda 1+1
replicate(3, 1+1)
## [1] 2 2 2

#pakartojame 10 kartų metimas() funkciją
replicate(10, metimas())
## [1] 10 8 5 6 6 4 4 7 4 5
```

1.4.3 Užduotis

1.4.3.1 Uždavinys

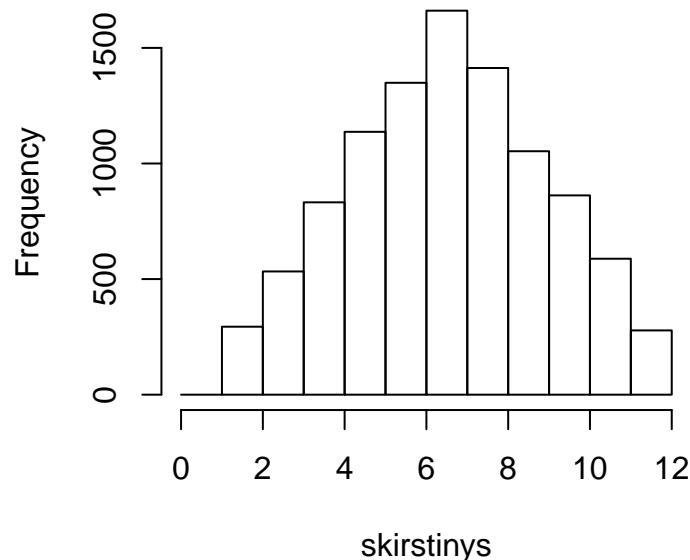
Jeigu būtume sąžiningas Casino, kaip atrodytų daugybės metimų skirstinys (metimas= 2 išmestų kauliukų suma)? Pasinaudodami `replicate()` funkcija, nubraižykite histogramą metimo sumų, jeigu funkcija metimas būtų pakartota 10 000 kartų

1.4.3.2 Sprendimas

```
# jeigu vidurkis vieno kauliuko:
mean(1:6)
## [1] 3.5
# tai dvejų kauliukų expected value (matematinė viltis) = vidurkis
2*mean(1:6)
## [1] 7
```

```
skirstinys <- replicate(10000, metimas())
hist(skirstinys, breaks = seq(from=0, to=12, b=1))
```

Histogram of skirstinys



Na bet būti sąžiningu Casino? Kur tai matyta... Ką reikia pakeisti?

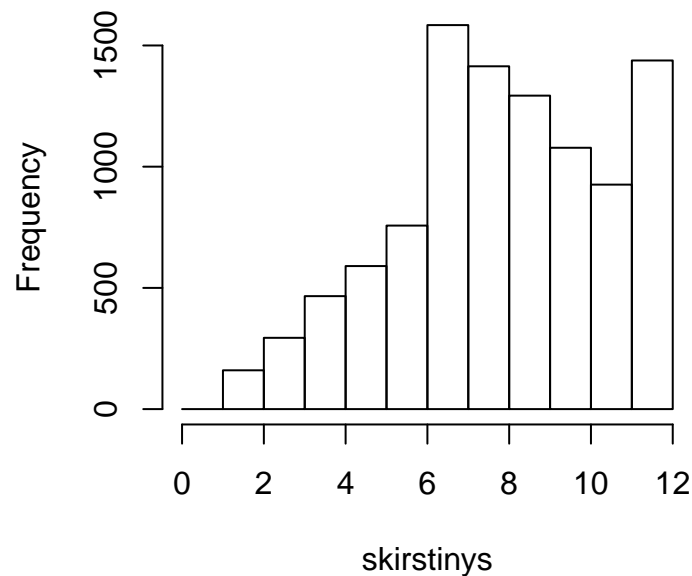
```
metimas2 <- function(){
  kauliukas <- 1:6
  kauliukai <- sample(kauliukas, size=2, replace = TRUE, prob = c(rep(1/8,5), 3/8))
  #prob = c(1/8,1/8,1/8,1/8,1/8, 3/8)
```

```
    sum(kauliukai) #svarbu iššaukti rezultatą  
}
```

Kaip atrodo tokių “pagerintų kauliukų” matematinės vilties skirstinys?

```
skirstinys <- replicate(10000, metimas2())  
hist(skirstinys, breaks = seq(from=0, to=12, b=1))
```

Histogram of skirstinys



```
mean(skirstinys)  
## [1] 8.2817
```

2 Black Jack

2.1 Įvadas

Šio užsiėmimo tikslai:

- pasikartoti skirtingus R objektus,
- subsetting,
- keisti reikmės objektuose

Šis užsiėmimas remiasi “Hands-On Programming with R” (Garett Golemund)