Duomenų analizės įvadas

2.2. dalis - R programavimas

Justas Mundeikis

VU EVAF

2019-03-16

Turinys

Loop funkcijos

Loop funkcijos

Rašant skriptus, `for`, `while` ir kiti loopai yra tinkami, bet jeigu norima parašyti kodą tiesiog konsolėje, tada susiduriama su daug problemų.

- `lapply`: loopina per list ir paleidžia funkciją kiekvienam elementui
- `sapply`: kaip ir `lapply` tik supaprastina rezultatus
- `apply`: taiko funkciją 'over the margins of an array'
- `tapply`: taiko funkciją vektoriaus dalims
- `mapply`: multivariatinė `lapply` versija

lapply

lapply

`lapply` priima 3 argumentus: (1) list objektą, (2) funkciją arba funkcijos pavadinimą, (3) galimus funkcijos papidlomus argumentus

Jeigu `X` nėra list, tada R bando paversti `X` list objektu.

```
args(lapply)
## function (X, FUN, ...)
## NULL
lapply
## function (X, FUN, ...)
## {
##     FUN <- match.fun(FUN)
##     if (!is.vector(X) || is.object(X))
##         X <- as.list(X)
##     .Internal(lapply(X, FUN))
## }
## <bytecode: 0x5566529da958>
## <environment: namespace:base>
```

lapply

`lapply` visad grąžina list klasės objektą

```
x <-  list(a=1:10, b=rnorm(10), c=seq(from=100, to=200, by=2))
lapply(x, mean)
## $a
## [1] 5.5
##
## $b
## [1] 0.2524767
##
## $c
## [1] 150
```

lapply

```
x <- 1:3
as.list(1:3)
## [[1]]
## [1] 1
##
## [[2]]
## [1] 2
##
## [[3]]
## [1] 3
lapply(x, runif)
## [[1]]
## [1] 0.6719481
##
## [[2]]
## [1] 0.4234126 0.2658082
##
## [[3]]
## [1] 0.4173505 0.2407724 0.6552593
```

lapply

Išnaudojant `...` galime perleisti papildomus argumentus `runif` funckijai:

```
x <- 1:3
as.list(1:3)
## [[1]]
## [1] 1
##
## [[2]]
## [1] 2
##
## [[3]]
## [1] 3
lapply(x, runif, min=5, max=10)
## [[1]]
## [1] 8.725853
##
## [[2]]
## [1] 8.815029 7.111812
##
## [[3]]
```

lapply

`lapply` ir kitos apply funkcijos gali naudotis anoniminėmis funkcijomis, t.y. niekur kitur nedefinuotomis funkcijomis

```
x <- list(a=matrix(1:9, nrow=3, ncol = 3), b=matrix(1:4, nrow = 2, ncol=2))
lapply(x, function(elt) elt[,1, drop=FALSE]) #elt yra anoniminė funkcija
## $a
##      [,1]
## [1,]    1
## [2,]    2
## [3,]    3
##
## $b
##      [,1]
## [1,]    1
## [2,]    2
```

sapply

sapply

`sapply` bando supaprastinti `lapply` rezultatus, jeigu įmanoma

- jeigu `lapply` grąžintu list, kurių kiekvienas elementas yra 1 ilgumo, tada `sapply` grąžina vektorių
- jeigu `lapply` grąžintu list, kurių kiekvienas elementas yra >1 ir vienodo ilgumo, tada `sapply` grąžina matricą
- jeigu netinka pirma du variantai, grąžina list

sapply

```
x <-  list(a=1:10, b=rnorm(10), c=seq(from=100, to=200, by=2))
lapply(x, mean)
## $a
## [1] 5.5
##
## $b
## [1] 0.4251241
##
## $c
## [1] 150
```

sapply

```
x <-  list(a=1:10, b=rnorm(10), c=seq(from=100, to=200, by=2))
sapply(x, mean)
##           a           b           c
##   5.5000000   0.1021237 150.0000000
```

apply

`apply` naudoajma taikyti funkcijas dataframe, matricų eilutėms ar stulpeliams. `apply` iš esmės supaprastine `for loop` naudijimą.

```
str(apply)
## function (X, MARGIN, FUN, ...)
```

apply

```
x <- matrix(1:4,2,2)
x
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
apply(x, 1, mean)
## [1] 2 3
apply(x, 2, mean)
## [1] 1.5 3.5
apply(x, 1, sum)
## [1] 4 6
apply(x, 2, sum)
## [1] 3 7
```

apply

Jeigu norima apskaičiuoti dataframe / matricų eilučių ar stuleplių sumas / vidurkius, galima naudoti jau supaprastinas funckijas, jos veikia dar greičiau, nei originalas.

```
rowSums=apply(x,1,sum) rowMeans=apply(x,1,mean)
colSums=apply(x,2,sum) colMeans=apply(x,2,mean)
```

apply

```
x <- matrix(rnorm(200),20,10)
apply(x, 1, quantile, probs=c(0.25 ,0.5, 0.75))
##            [,1]       [,2]       [,3]        [,4]       [,5]         [,6]
## 25% -1.1004017 -0.8547683 -1.2813248 -0.64945572 -0.2201941 -1.07191246
## 50%  0.1221921 -0.1906842  0.2143838 -0.09093318  0.2011061 -1.01919673
## 75%  0.5222819  0.5919317  0.8878092  0.23082686  0.7551597  0.09217805
##            [,7]       [,8]       [,9]      [,10]      [,11]       [,12]
## 25% -0.1331935 -0.2481558 -1.4190440 -1.1612452 -1.2993890 -1.0763143
## 50%  0.4863995  0.1885003 -0.1680950 -0.4400157 -0.9043489 -0.3308245
## 75%  0.5678069  0.8258982  0.3059019  0.5369320 -0.1634602 -0.0295602
##           [,13]        [,14]      [,15]       [,16]      [,17]        [,18
## 25% -0.3943433 -0.945413692 -1.5419104 -0.97948907 -1.0413642 -0.7838835
## 50%  0.2711753  0.003369702 -0.3199503 -0.54684740 -0.1477208 -0.0948046
## 75%  0.9172764  0.611524161  0.6634889 -0.07339726  0.4383544  0.5979586
##           [,19]      [,20]
## 25% -0.3208434 -0.9617296
## 50%  0.1115974 -0.1689614
## 75%  0.5849055  0.1932603
```

apply

```
x <- array(data=rnorm(40), dim = c(2,2,10))
apply(x, c(1,2), mean)
##             [,1]        [,2]
## [1,] -0.3875616  0.29571962
## [2,] -0.2949055 -0.02082904
```

mapply

mapply

`mappl` taiko paraleliai (vienu metu) funckiją skirtingiems argumentams

```
str(mapply)
## function (FUN, ..., MoreArgs = NULL, SIMPLIFY = TRUE, USE.NAMES = TRUE)
```

- `FUN` yra funkcija, kuri bus taikoma
- `...` argumentai, kuriais naudojamasi funkcijoje
- `MoreArgs` kiti `FUN` argumentai
- `SIMPLIFY` ar rezultatas turėtų būti simplifikuotas kaip `sapply`

mapply

```
list(rep(1,4), rep(2,3), rep(3,2), rep(4,1))
## [[1]]
## [1] 1 1 1 1
##
## [[2]]
## [1] 2 2 2
##
## [[3]]
## [1] 3 3
##
## [[4]]
## [1] 4
```

mapply

```
mapply(rep, 1:4, 4:1)
## [[1]]
## [1] 1 1 1 1
##
## [[2]]
## [1] 2 2 2
##
## [[3]]
## [1] 3 3
##
## [[4]]
## [1] 4
```

mapply

```
noise <- function(n, mean, sd){
        rnorm(n, mean, sd)
}

noise(4,1,2)
## [1]  0.815746  1.575084 -1.330642  1.411042
noise(1:4,1:4,0.01)
## [1] 0.994897 2.003265 2.998282 3.998115
```

mapply

```
noise <- function(n, mean, sd){
        rnorm(n, mean, sd)
}

mapply(noise, 1:4, 1:4, 0.1)
## [[1]]
## [1] 1.013708
##
## [[2]]
## [1] 2.078110 2.071801
##
## [[3]]
## [1] 2.998873 2.937595 3.046855
##
## [[4]]
## [1] 4.104642 4.047430 3.998437 4.103961
# list( noise(1,1,0.1),noise(2,2,0.1),noise(3,3,0.1),noise(4,4,0.1)))
```

tapply

tapply

```
str(tapply)
## function (X, INDEX, FUN = NULL, ..., default = NA, simplify = TRUE)
```

- `X` yra vektorius
- `INDEX` faktorius arba faktorių list
- `FUN` taikoma funkcija
- `...` papildomi `FUN` argumentai
- `simplify` ar supaprastinti rezultatus

tapply

```
x <- c(rnorm(10), runif(10), rnorm(10,1))
x
##  [1] -0.7231060  1.1326129  1.0512988 -0.2689475  0.1850268  0.2023755
##  [7]  0.7552277  1.1811535 -0.2585635  0.6071436  0.2637104  0.1753065
## [13]  0.5591984  0.8262232  0.9467480  0.7652819  0.4253932  0.5710458
## [19]  0.9770959  0.9933834  0.2724219  1.3371940  1.1002621 -0.9620210
## [25]  0.3119945  3.7750138  1.8188418  1.2528800 -0.4404487 -0.3399938
# Generate factors by specifying the pattern of their levels.
#gl(n, k, length = n*k, labels = seq_len(n), ordered = FALSE)
f <- gl(3,10)
f
##  [1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3
## Levels: 1 2 3
tapply(x, f, mean)
##         1         2         3
## 0.3864222 0.6503387 0.8126144
```

tapply

```
tapply(x, f, mean, simplify = FALSE)
## $`1`
## [1] 0.3864222
##
## $`2`
## [1] 0.6503387
##
## $`3`
## [1] 0.8126144
```

tapply

```
tapply(x, f, summary)
## $`1`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.7231 -0.1477  0.4048  0.3864  0.9773  1.1812
##
## $`2`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1753  0.4588  0.6682  0.6503  0.9166  0.9934
##
## $`3`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.9620 -0.1869  0.7061  0.8126  1.3161  3.7750
```

split

`split` padalina vektorių arba kitą objektą į grupes priklausomai nuo faktorių arba faktorių list

```
str(split)
## function (x, f, drop = FALSE, ...)
```

- `x` vektorius / list / dataframe
- `f` faktorius arba faktorių list
- `drop` indikuoja, ar tušti faktoriai turėtų būti panaikinti

split

split

```
x <- c(rnorm(10), runif(10), rnorm(10,1))
f <- gl(3,10)
split(x, f)
## $`1`
##  [1] -0.33028847  1.53207684 -0.74501340 -1.48387865 -0.29717694
##  [6]  0.21877092  0.16867412  0.46984638  0.02587648  1.29212789
##
## $`2`
##  [1] 0.99327997 0.18548999 0.05223415 0.47748168 0.22374088 0.74541531
##  [7] 0.08026931 0.40778336 0.30331372 0.98489960
##
## $`3`
##  [1]  0.74640808 -0.06320149  1.09042925  1.65383843  1.04352568
##  [6]  1.36306360 -0.87970006  1.43736967  0.75167612  0.49934197
# dabar galima naudoti lapply / sapply
```

split

```
lapply(split(x, f),mean)
## $`1`
## [1] 0.08510152
##
## $`2`
## [1] 0.4453908
##
## $`3`
## [1] 0.7642751
tapply(x, f, mean)
##          1          2          3
## 0.08510152 0.44539080 0.76427513
```

split

```
head(airquality)
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
## 6    28      NA 14.9   66     5   6
```

split

```
s <- split(airquality, airquality$Month)
lapply(s, function(x) colMeans(x[,1:4]))
## $`5`
##    Ozone  Solar.R     Wind     Temp
##       NA       NA 11.62258 65.54839
##
## $`6`
##     Ozone   Solar.R      Wind      Temp
##        NA 190.16667  10.26667  79.10000
##
## $`7`
##      Ozone    Solar.R       Wind       Temp
##         NA 216.483871   8.941935  83.903226
##
## $`8`
##     Ozone   Solar.R      Wind      Temp
##        NA        NA  8.793548 83.967742
##
## $`9`
##    Ozone  Solar.R     Wind     Temp
```

split

```
s <- split(airquality, airquality$Month)
lapply(s, function(x) colMeans(x[,1:4], na.rm=TRUE))
## $`5`
##     Ozone   Solar.R      Wind      Temp
##  23.61538 181.29630  11.62258  65.54839
##
## $`6`
##     Ozone   Solar.R      Wind      Temp
##  29.44444 190.16667  10.26667  79.10000
##
## $`7`
##      Ozone    Solar.R       Wind       Temp
##  59.115385 216.483871   8.941935  83.903226
##
## $`8`
##      Ozone    Solar.R       Wind       Temp
##  59.961538 171.857143   8.793548  83.967742
##
## $`9`
##     Ozone   Solar.R      Wind      Temp
```

split

```
s <- split(airquality, airquality$Month)
sapply(s, function(x) colMeans(x[,1:4], na.rm=TRUE))
##                 5         6          7          8         9
## Ozone    23.61538  29.44444  59.115385  59.961538  31.44828
## Solar.R 181.29630 190.16667 216.483871 171.857143 167.43333
## Wind     11.62258  10.26667   8.941935   8.793548  10.18000
## Temp     65.54839  79.10000  83.903226  83.967742  76.90000
```

split

```
regionas <- as.factor(rep(c("Vilnius", "Kaunas", "Klaidpėda"), each=10 ))
lytis <- as.factor(c("M", "V"))
x <- data.frame(metai=rep(2013:2017),
                regionas=rep(c("Vilnius", "Kaunas", "Klaidpėda"), each=10 ),
                lytis=c("M", "V"),
                bvp=rep(runif(5,100,200),3),
                vartojimas=rnorm(30)
                )
s <- (split(x, list(regionas, lytis)))
```

split

```
head(x,15)
##    metai regionas lytis      bvp vartojimas
## 1   2013  Vilnius     M 182.1965  0.1374447
## 2   2014  Vilnius     V 107.5730  0.3819673
## 3   2015  Vilnius     M 143.6011 -0.5554019
## 4   2016  Vilnius     V 160.6046  0.4907543
## 5   2017  Vilnius     M 192.9062  1.1916157
## 6   2013  Vilnius     V 182.1965  0.4028319
## 7   2014  Vilnius     M 107.5730 -0.1444782
## 8   2015  Vilnius     V 143.6011  1.3912334
## 9   2016  Vilnius     M 160.6046 -0.6379880
## 10  2017  Vilnius     V 192.9062 -0.2452594
## 11  2013   Kaunas     M 182.1965  0.4925920
## 12  2014   Kaunas     V 107.5730 -0.7312743
## 13  2015   Kaunas     M 143.6011 -0.7074130
## 14  2016   Kaunas     V 160.6046 -0.8076628
## 15  2017   Kaunas     M 192.9062  0.2572263
```

split

```
head(s,2)
## $Kaunas.M
##    metai regionas lytis      bvp vartojimas
## 11  2013   Kaunas     M 182.1965  0.4925920
## 13  2015   Kaunas     M 143.6011 -0.7074130
## 15  2017   Kaunas     M 192.9062  0.2572263
## 17  2014   Kaunas     M 107.5730 -0.1033139
## 19  2016   Kaunas     M 160.6046 -0.2459296
##
## $Klaidpėda.M
##    metai  regionas lytis      bvp vartojimas
## 21  2013 Klaidpėda     M 182.1965 -0.1987318
## 23  2015 Klaidpėda     M 143.6011 -0.8170531
## 25  2017 Klaidpėda     M 192.9062  1.6554127
## 27  2014 Klaidpėda     M 107.5730 -0.4579310
## 29  2016 Klaidpėda     M 160.6046  1.9531693
```

split

```
sapply(s, function(x) mean(x[,4]))
##    Kaunas.M Klaidpėda.M   Vilnius.M    Kaunas.V Klaidpėda.V   Vilnius.V
##    157.3763    157.3763    157.3763    157.3763    157.3763    157.3763
sapply(s, function(x) colMeans(x[,4:5]))
##                Kaunas.M Klaidpėda.M     Vilnius.M    Kaunas.V Klaidpėda.
## bvp        157.37628375 157.3762837 157.376283747 157.3762837 157.376283
## vartojimas  -0.06136762   0.4269732  -0.001761549  -0.1475587  -0.579146
##               Vilnius.V
## bvp        157.3762837
## vartojimas   0.4843055
```

Distribucijos

Distribucijos

Ne retai atliekant įvarius tyrimus ar skaičiuojant tikimybes statistikoje, reikės remtis tam tikrais skirstiniais. R gali generuoti įvarius skirstinius (*distributions*) `?distributions`

- dnorm
- dgamma
- beta
- dpois

ir t.t.

Distribucijos

Šioje dalyje aptarsime

- Binomial Distribution
- Poisson Distribution
- Continuous Uniform Distribution
- Exponential Distribution
- Normal Distribution

Distribucijos

Visos distribucijos galimos su 4 funkcijomis:

```
# ?dnorm

dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)
```

- d density
- r random number generation
- p cumulative distribution
- q quantile function

set.seed(...)

Ypatingai savo tyrimuose, jeigu naudojant sugeneruotus atsitiktinius skaičius iš tam tikro skirstinio, būtina naudoti `set.seed()`, tam, jog tyrimas būtų atkartojamas.

```
set.seed(1)
rnorm(n=5, mean=5, sd=2)
## [1] 3.747092 5.367287 3.328743 8.190562 5.659016
rnorm(n=5, mean=5, sd=2)
## [1] 3.359063 5.974858 6.476649 6.151563 4.389223
set.seed(1)
rnorm(n=5, mean=5, sd=2)
## [1] 3.747092 5.367287 3.328743 8.190562 5.659016
```

Binominis skirstinys

Dichotomine matavimų skale matuojamų požymių reikšmių skirstinys. Skirstinys yra diskretus ir apibūdinamas parametrais n ir p. Parametras $n \geq 0$ reiškia bandymų skaičių, o p – požymio tikimybę įgyti vieną iš dviejų galimų reikšmių.

Binominio skirstinio pasiskirstymo tankio funkcija (tikimybė gauti x reikmšę su n bandymų ir p tikimybės reikmše):

$$f(x) = \binom{n}{x} p^x (1-p)^{n-x} \text{ kur } x = 1, 2, 3, ..., n$$

Binomial distribution

Binomial distribution

Tarkime duomenų analizės teste yra 10 klausimų, kurių kiekvienas turi 4 galimus atsakymus, iš kurių tik vienas yra teisingas. Tarkime studentas atėjo visiškai nepasiruošęs ir visiškai atsitiktinai apsirinks atsakamus. Norint išlaikyti testą, reikia teisingai astakyti į ne mažiau kaip 5 klausimus. Kokia tikimybė, jog studentas neišlaikys testo?

- p = 1/4=0.25 ir (1-p)=1-0.25=0.75
- n = 10
- x=4

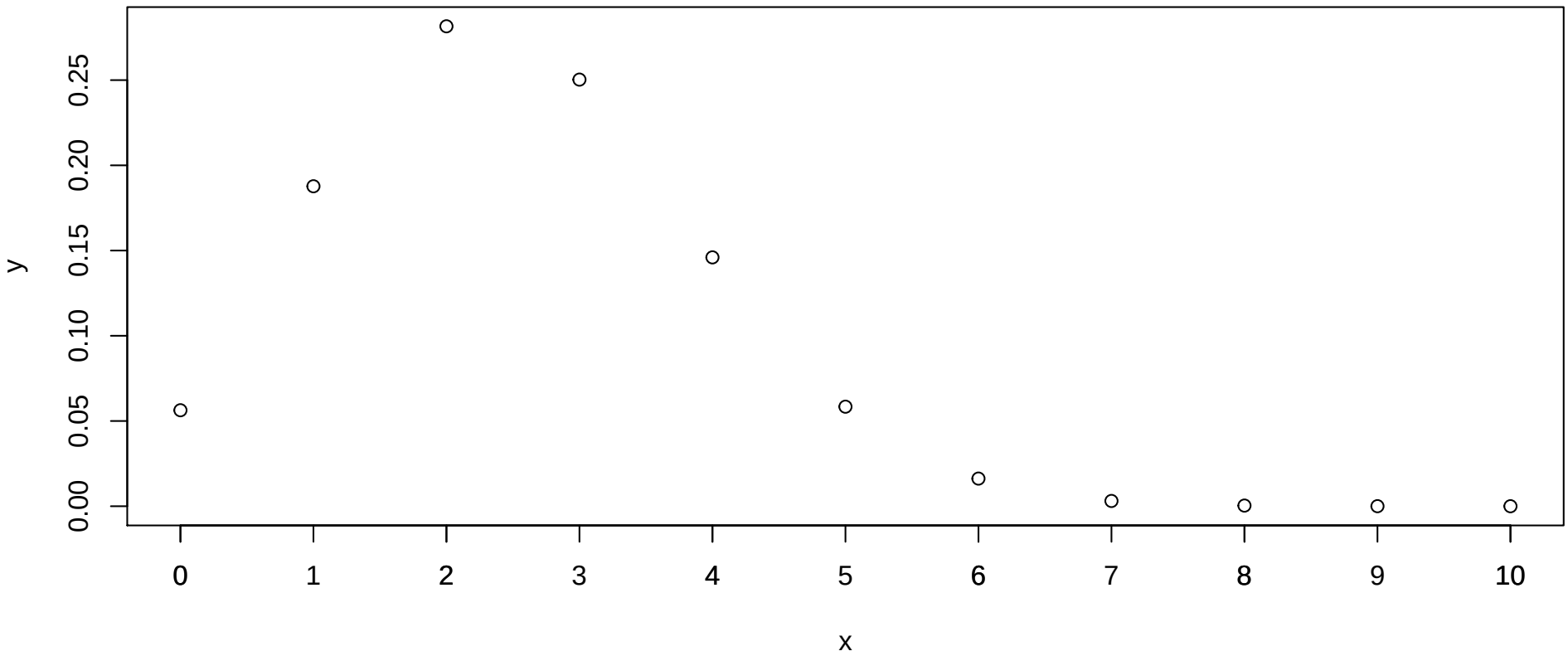
Binomial distribution

- p = 1/4=0.25 ir (1-p)=1-0.25=0.75
- n = 10
- x=4

```
# tikimybė jog studentas atsakys lygiai 4 teisingai
dbinom(x=4, size = 10, prob = 0.25)
## [1] 0.145998
```

Binomial distribution

```
x <- seq(from=0, to=10, by=1)
y <- dbinom(x, size=10, prob=0.25)
plot(x,y, type = "p")
axis(side = 1, at = x,labels = T)
```



Binomial distribution

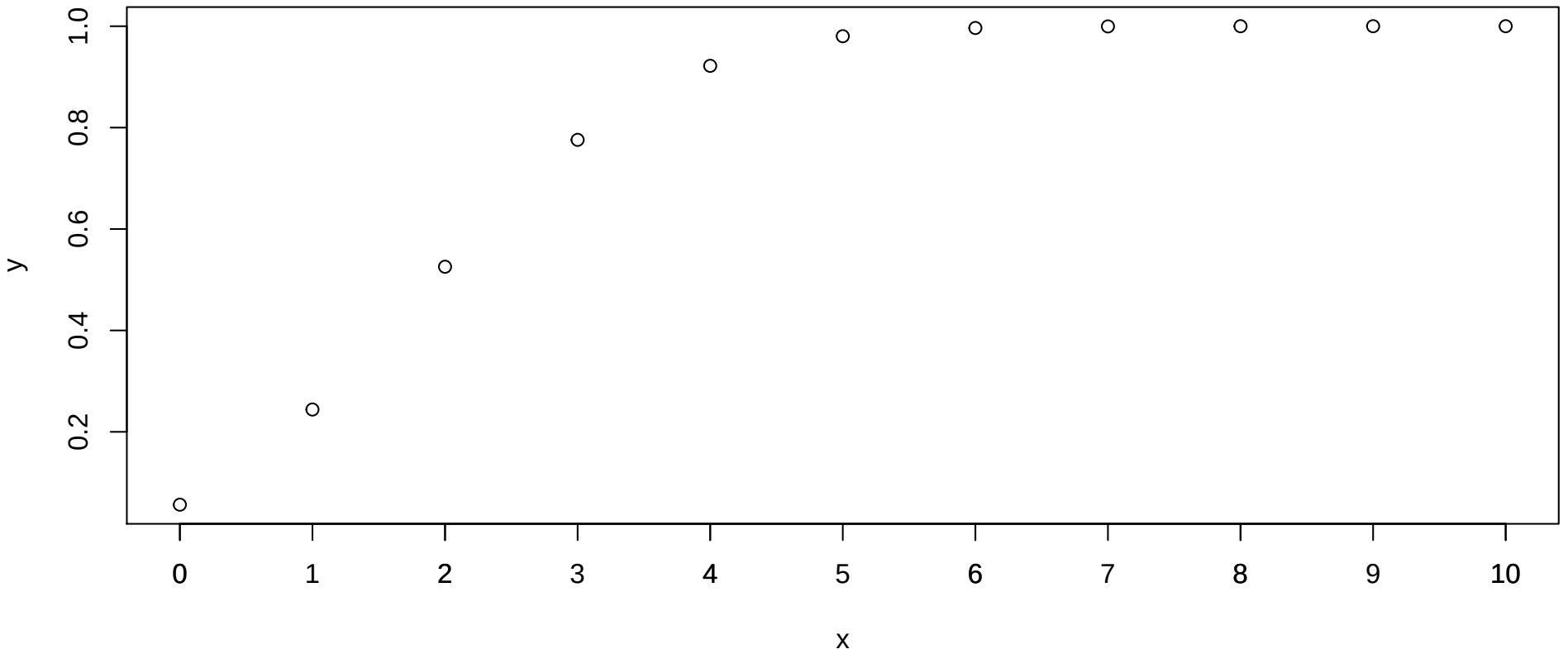
tačiau norint žinoti visas vertes iki 4

```
# todėl norint žinoti tikimybę jog studentas atsakys į 4 arba mažiau
dbinom(x=0, size = 10, prob = 0.25)+
        dbinom(x=1, size = 10, prob = 0.25)+
        dbinom(x=2, size = 10, prob = 0.25)+
        dbinom(x=3, size = 10, prob = 0.25)+
        dbinom(x=4, size = 10, prob = 0.25)
## [1] 0.9218731

# alternatyviai galima pasinaudoti pbinom()
pbinom(q=4, size= 10, prob = 0.25, lower.tail = TRUE)
## [1] 0.9218731
# tačiau piktąjį dėstytoją domina,
# kokia tikimybė, jog studentas "praslys":
pbinom(q=4, size= 10, prob = 0.25, lower.tail = FALSE)
## [1] 0.07812691
```

Binomial distribution

```
x <- seq(from=0, to=10, by=1)
y <- pbinom(x, size=10, prob=0.25)
plot(x,y, type = "p")
axis(side = 1, at = x,labels = T)
```



Binomial distribution

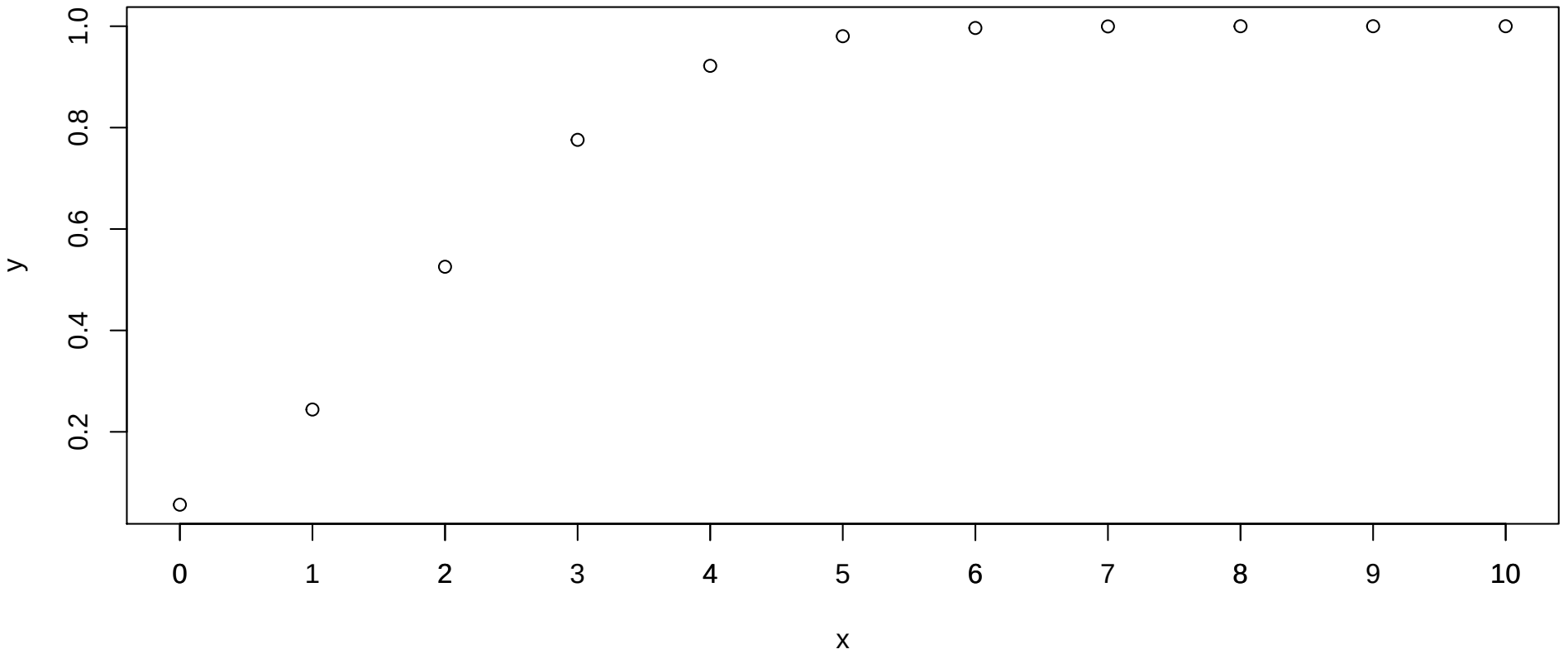
Tarkime dėstytojas nori nustatyti ribą, i kiek klausimų turi teisingai atsakyti studentai, kai:

- studentai turėdami 4 galimus pasirinkimus (daugiau alternatyvių atsakymų dėstytojas nenori sugalvoti, nes tingi)
- destyjas nenori, kad studentai praslystų pro testą didesne nei 10% tikimybe
- destytojas tingi galvoti daugiau nei 10 klausimų

```
qbinom(0.1, 10, 0.25, lower.tail = FALSE)
## [1] 4
```

Binomial distribution

```
x <- seq(from=0, to=10, by=1)
y <- pbinom(x, size=10, prob=0.25)
plot(x,y, type = "p")
axis(side = 1, at = x,labels = T)
```



Poisson distribution

Poisson distribution

Dichotomine matavimų skale matuojamų požymių reikšmių skirstinys. Skirstinys yra diskretus ir apibūdinamas parametrais n ir p. Parametras $n \geq 0$ reiškia bandymų skaičių, o p – požymio tikimybę įgyti vieną iš dviejų galimų reikšmių.

Binominio skirstinio pasiskirstymo tankio funkcija (tikimybė gauti x reikmšę su n bandymų ir p tikimybės reikmše):

$$f(x) = \frac{\lambda^x e^{-\lambda}}{x!} \text{ kur } x = 1, 2, 3, ..., n$$

Poisson distribution

Poisson distribucija

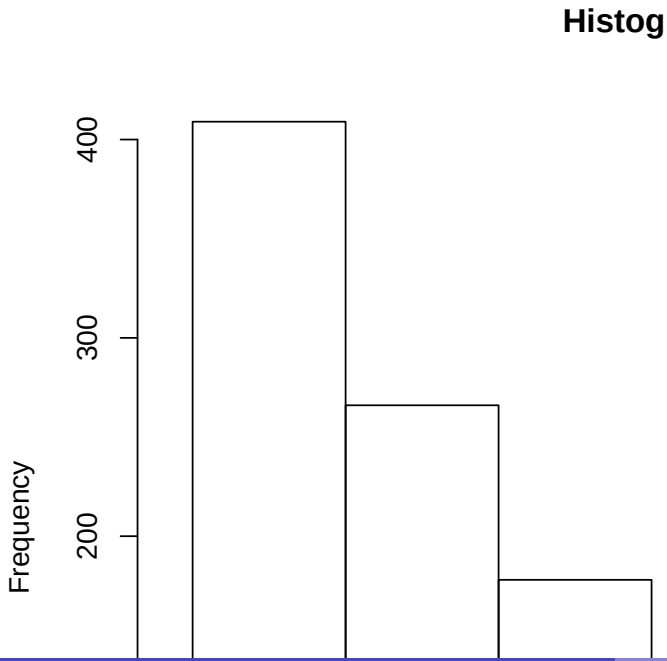
```
# ?dpois

dpois(x, lambda, log = FALSE)
ppois(q, lambda, lower.tail = TRUE, log.p = FALSE)
qpois(p, lambda, lower.tail = TRUE, log.p = FALSE)
rpois(n, lambda)
```

Poisson distribution

Poisson distribucija, kur λ yra vidutinė įvykio tikimybė per tam tikrą laikotarpį

```
rpois(n=10, lambda = 1)
##  [1] 0 0 1 1 2 1 1 4 1 2
rpois(n=10, lambda=2)
##  [1] 4 1 2 0 1 1 0 1 4 1
hist(rpois(n=1000, lambda=2))
```



Poisson distribution

Poisson distribucija, kur λ yra vidutinė įvykio tikimybė per tam tikrą laikotarpį.

Skambučių centras per valandą sulaukia 50 skambučių. *Maximum capacity* yra 65 skambučiai per valandą. Tada skambučiai nukreipiami į alternatyvų skambučių centrą, kuriame dirba beždžionėlės, tad klientai visad lieka nepatenkinti.

Klausimas, kokia yra tikimybė, jog per sekančią valandą skambučių centras sulauks: 5, 30, 60 (arba mažiau skambučių):

```
dpois(5, 50) ## 5 Pr(x=5), lambda=50
## [1] 5.022786e-16
dpois(30 , 50) ## 30  Pr(x=30), lambda=50
## [1] 0.0006771985
dpois(60, 50) ## 60  Pr(x=50), lambda=50
## [1] 0.02010487

ppois(5, 50) ## 5 arba mažiau skambučių Pr(x<=5), lambda=50
## [1] 5.567756e-16
```

Poisson distribution

Kokia tikimybė, jog skambučių centras sulauks daugiau nei skabučių centro maksimalus limitas? Jeigu įmonės išsikeltas tiklsas, jog nepatenkintų klientų būtų mažiau nei 1%, ar patartumėte vadovybei plėsti skambučių centro galimybes? Kiek papildomų darbuotjų reikai nusamdyti skambučiui centrui, jeigu 1 darbuotojas gali priimti 5 skambučius?

```
ppois(q=65, lambda = 50, lower.tail = TRUE)
## [1] 0.9827354
ppois(q=65, lambda = 50, lower.tail = FALSE)
## [1] 0.01726457

# Kiek papildomų skambučių reiktų papildomai galėti priimti?
qpois(p=0.01, lambda = 50, lower.tail = FALSE)
## [1] 67
ceiling(
        (qpois(p=0.01, lambda = 50, lower.tail = FALSE) - 50) / 5
        )
## [1] 4
```

Continuous uniform distribution

Exponential distribution

Normal distribution