Duomenų analizės įvadas

2-1 Dalis

Justas Mundeikis

2019 m. vasario 28 d.

2.1. Dalies turinys

R Istorija

- R yra S kalbos dialektas
- S kalba parašyta John Chambers et al. @Bell Labs 1976m.
- S buvo perrašyta 1988m. (v3) ir tapo labiau panaši į statistinę programą, o 1998m. išleista v4.
- R sukurtas 1991m. mokslinio darbo rėmuose (Ross Ihaka ir Robert Gentleman) Auckland universitete (Naujoji Zealandija)
- 1993m. R pristatytas visuomenei
- 1995m. R gavo GNU licenziją

R Istorija

- R sintaksė labai panaši į S
- R veikia ant su bet kokia operacine sistema
- Didelė bendruomenė, todėl labai daug paketų ir dažni bugfix'ai
- Santykinai lengva atlikti statistines analizes, tačiau suteikia beveik neribotas galimybes norintiems programuoti savo paketus
- R yra laisva programa (*free software*) remiantis GNU Public License

Free software

Jeigu kalbame apie "free software" turima omenyje 4 laisves

- 0. Laisvė naudotis programa, bet kuriuo tikslu
- 1. Laisvė analizuoti ir keisti programą pagal savo poreikius
- 2. Laisvė dalintis programos kopijomis
- 3. Laisvė dalintis pagerintomis kopijomis

1 ir 3 laisvėms būtina laisva prieiga prie programos kodo.
Philosophy of the GNU Project

R minusai

- R remiasi 40 metų senumo programa, todėl trūksta 3D grafikų
- Paketus kuria patys vartotojai, todėl jeigu nėra jau sukurto reikiamo funkcionalumo, reikia kurti pačiam
- Visi objektai R turi būti įkeliami į darbinę atmintį
- R nėra labai universali kalba

R ir RStudio instaliavimas

- R reikia instaliuoti iš CRAN
- https://cran.r-project.org/
- Paleidžiame R
- Tam kad būtų lengviau dirbti su R, turėti aibę papildomų funkcijų, instaliuojame RStudio
- https://www.rstudio.com/products/rstudio/download/
- Startuojame RStudio

R sistema

R susideda iš dviejų komponentų:

1. Bazinė R sistema su standartiniais paketais: stats, graphics, grDevices, utils, datasets, methods, base (galima pamatyti įvedus komandą search())
2. Visų kitų paketų

- dauguma R paketų saugomi CRAN (Comprehensive R Archive Network), iš kur atsisiunčiamas ir pats R
- available.packages() funkcija, kuri surenką visą informaciją apie ezistuojančius R paketus @CRAN

```
a <- available.packages()
length(a)
```

- Šiuo metu : 230180 paketai
- Taip pat galima instaliuoti ir paketus esančius @GitHub

Kur rasti pagalbą

99.99%, na ką jau, 150% visi susidursite su problemomis, kai kažkas neveiks kaip norite, kai R praneš apie klaidas ir t.t. ir kai patys nežinosite ką daryti toliau. Todėl toks "pagalbos eiliškumas":

1. R, R paktetų, Git, GitHub dokumentacija, "help" funkcija
2. Google (Ctrl+C Ctrl+V error code) 99.99% kažkas jau tą problemą turėjo
3. Kursiokai / Mokslo grupė
4. https://stackoverflow.com/ (žr. sekanti skaidrė)
5. Dėstytojas (žr. sekanti skaidrė)

Stackoverflow

- https://stackoverflow.com/
- Kokius konkrečiai žingsnius atlikote
- Kokio rezultato tikitės
- Kokį rezultatą gaunate
- Kokią R versiją, kokius paketus naudojate (retai: kokia operacinė sistema)
- Visada geriausia aprašyti problemą, bei pateikti visą kodą, leidžiantį atkartoti Jūsų problemą
- Antraštė turėtų būti trumpa ir aiški

R Input Output

"<-" yra priskyrimo operatorius, ">" promt (CLI buvo $)

```
> x <- 1
> print(x)
[1] 1
> msg <- "hello world"
> print(msg)
[1] "hello world"
```

Komentarai atskiriami su # viskas į dešinę nuo # ignoruojama (toje eilutėje)

```
> msg <- "hello world" #pirma žinute
> msg # autoprint prints values without entering command print()
[1] "hello world"
> x <-
+
```

ESC arba pabaigti įvesti. [1] indikuoja vektoriaus reikšmės numerį

R rezervuoti objektai

- ?reserved komanda parodo, kokie objektų pavadinimai negali būti perrašyti
- nebent naudojamos backticks kabutes pvz.: ‘if‘

if	else	repeat	while	function
for	in	next	break	TRUE
FALSE	NULL	Inf	NaN	NA
NA_integer_	NA_real_	NA_complex_	NA_character_	...

R aritmetikos operatoriai

Operator	**Description**
+	Addition
–	Subtraction
*	Multiplication
/	Division
^	Exponent
%%	Modulus (Remainder from division)
%/%	Integer Division

R Relational Operators

Operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
Err:520	Equal to
!=	Not equal to

R loginiai operatoriai

Operator	Description
!	Logical NOT
&	Element-wise logical AND
&&	Logical AND
|	Element-wise logical OR
||	Logical OR

R loginiai operatoriai

```
v <- c(3,1,TRUE,2+3i); t <- c(4,1,FALSE,2+3i)

It is called Element-wise Logical AND operator. It combines each ↩
    element of the first vector with the corresponding element of↩
     the second vector and gives a output TRUE if both the ↩
    elements are TRUE.

print(v&t)

[1]  TRUE  TRUE FALSE  TRUE

It is called Element-wise Logical OR operator. It combines each ↩
    element of the first vector with the corresponding element of↩
     the second vector and gives a output TRUE if one the ↩
    elements is TRUE.

print(v|t)

[1]  TRUE FALSE  TRUE  TRUE
```

R loginiai operatoriai

```
v <- c(3,0,TRUE,2+2i)
t <- c(1,3,TRUE,2+3i)

It is called Logical NOT operator. Takes each element of the ↩
    vector and gives the opposite logical value.

print(!v)
[1] FALSE  TRUE FALSE FALSE

Called Logical AND operator. Takes first element of both the ↩
    vectors and gives the TRUE only if both are TRUE.

print(v&&t)
[1] TRUE

Called Logical OR operator. Takes first element of both the ↩
    vectors and gives the TRUE if one of them is TRUE.

print(v||t)
[1] FALSE
```

R Input Output

Operatorius : sukuria eiles (sequence)

```
> x <- 1:30
> x
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
[19] 19 20 21 22 23 24 25 26 27 28 29 30
```

R objektai

R turi 5 bazinius objektų tipus / klases (*atomic classes*):

- numeric (double): 1, 4.5, -1.1...
- integer 1L, 2L, 3L (sveikas skaičius)
- complex 1i , 2i, 3i
- character: "vilnius", "amžius" (visada su kabutėmis)
- logical TRUE /FALSE arba T/F
- su komanda typeof() galima pasitikrinti klasę

Dažniausiai naudojamas vektorius, kuriame gali būti tik tos pačios klasės objektai. Tuščią vektorių galima sukurti su komanda vector()
list (sąrašas) gali talpinti įvairių klasių objektus.

R objektai

- Skaičius R supranta kaip numeric klasės objektus
- Jeigu reikia pilno skaičiaus (integer) tada skaičių reikia pabaigti su L raide
- Inf suprantamas kaip begalybė
- NAN ("not a number"), arba trūkstama reikšmė

```
> x<-2L
> x
[1] 2
> x <-2.1L
Warning message:
integer literal 2.1L contains decimal; using numeric value
> Inf
[1] Inf
> 1/Inf
[1] 0
> 0/0
[1] NaN
```

Atributai

R objektai gali turėti atributus.

- names, dimnames
- dimensions (e.g matricos)
- class (numeric, charackter)
- length
- kiti vartotojo priskirti atributai
- attributes() leidžia nustatyti / keisti objekto atributus

Vektorių sukūrimas

- c() funkcija leidžia sukurti objektų vektorius
- c() iš concatenate

```
> x <- c(0.2 , 0.6) # numeric class
> x <- c(TRUE, FALSE) #logical class
> x <- c(T, F) #logical class
> x <- c("a", "b", "c") #character class
> x <- 1:5  #integer class
> x <- c(1+0i, 2+4i) #complex class
```

- galima sukurti tuščią vektorių, nurodant kokios klasės objektai jame bus ir kokia vektoriaus dimensija

```
> x <- vector(mode="numeric", length = 8)
> x
[1] 0 0 0 0 0 0 0 0
```

Vektorių sujungimas (*coersion*)

- jeigu su c() sujungiami skirtingų klasių objektai, R priskiria bendriausią klasę visiems vektoriuje esantiems objektams

```
> x <- c(0.2 , "a") # character class
> x <- c(TRUE, FALSE, 3) #numeric class
> x <- c("a", TRUE, FALSE) #character class
```

- TRUE=1, FALSE=0
- procesas kuris vyksta vadinamas *coersion*

Vektorių sukūrimas

- Vektorius galima rankiniu būdu priskirti tam tikrai klasei

```
> x<-10:19
> class(x)
[1] "integer"
> as.numeric(x)
 [1]  0  1  2  3  4  5  6  7  8  9 10
> as.logical(x)
 [1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[10]  TRUE  TRUE
> as.character(x)
 [1] "0"  "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"
[11] "10"
> as.factor(x)
 [1] 0  1  2  3  4  5  6  7  8  9  10
Levels: 0 1 2 3 4 5 6 7 8 9 10
```

Vektorių sukūrimas

- tačiau nelogiški manualūs priskyrimai generuos NAs

```
> x<- c("a", "b", "c")
> as.numeric(x)
[1] NA NA NA
Warning message:
NAs introduced by coercion

> as.logical(x)
[1] NA NA NA

> as.complex(x)
[1] NA NA NA
Warning message:
NAs introduced by coercion
```

List - sąrašas

- List gali talpinti įvairių klasių objektus

```
> x <- list(1.2, 3L, TRUE, F, 1+4i, 1:3)
> x
[[1]]
[1] 1.2
[[2]]
[1] 3
[[3]]
[1] TRUE
[[4]]
[1] FALSE
[[5]]
[1] 1+4i
[[6]]
[1] 1 2 3
```

- [[*nr*]] nurodo list objekto numerį

Matricos

- Matricos, tai tas pats vektoriaus objektas, tačiau turintis dimensijos nustatymus

```
> x <- matrix(nrow = 3, ncol = 3)
> x
     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
[3,]   NA   NA   NA

> dim(x)
[1] 3 3

> attributes(x)
$dim
[1] 3 3
```

Matricos

- Matricos užpildomos stulpeliniu būdų, jeigu nenurodoma kitaip
- ?matrix parodo funkcijos manual

```
> m <- matrix(1:9, nrow = 3, ncol = 3)
> m
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

>? matrix

> m <- matrix(1:9, nrow = 3, ncol = 3, byrow = TRUE)
> m
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

Matricos

- Vektorius be dimensijų yra paprastas vektorius
- Tačiau vektoriui galima suteikti dimensijas post factum, tada vektorius tampa matrica

```
> v <- 1:12
> v
 [1]  1  2  3  4  5  6  7  8  9 10 11 12
> dim(v) <-c(4,3)
> v
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
```

cbind , rbind

- cbind (columnbind) ir rbind (rowbind) iš atskirų vektorių sukuria matricas

```
> x <-1:3
> y <- 20:22
> cbind(x,y)
      x  y
[1,] 1 20
[2,] 2 21
[3,] 3 22
> rbind(x,y)
  [,1] [,2] [,3]
x    1    2    3
y   20   21   22
```

cbind , rbind

- Tačiau jeigu vektorių dydis ne toks pats... r coersion'a

```
> x <-1:3
> y <- 1:5
> cbind(x,y)
      x y
[1,] 1 1
[2,] 2 2
[3,] 3 3
[4,] 1 4
[5,] 2 5
Warning message:
In cbind(x, y) : number of rows of result is not a multiple ↩
    of vector length (arg 1)
> rbind(x,y)
  [,1] [,2] [,3] [,4] [,5]
x    1    2    3    1    2
y    1    2    3    4    5
Warning message:
In rbind(x, y) :number of columns of result is not a ↩
    multiple of vector length (arg 1)
```

Faktoriai

- Faktorių klasė skirta kategoriniams kintamiesiems (vardiniai, ranginiai)
- Faktoriai yra svarbūs modeliuojant bei kartais grafikams

```
> x <-factor(c("taip", "ne", "taip", "taip", "ne"))
> x
[1] taip ne   taip taip ne
Levels: ne taip
> table(x)
x
  ne taip
   2    3
> unclass(x)
[1] 2 1 2 2 1
attr(,"levels")
[1] "ne"   "taip"
```

Faktoriai

- Lygiai pagal alfabetinį eiliškumą pasirodantį vektoriuje, arba nurodoma manualiai

```
> x <-factor(c("girtas", "blaivus", "girtas" , "girtas" , "↩
    blaivus"),levels=c("girtas", "blaivus"))
> x
[1] girtas  blaivus girtas  girtas  blaivus
Levels: girtas blaivus
> table(x)
x
 girtas blaivus
      3       2
```

Trūkstami skaičiai

- Trūkstami skaičiai pateikiami kaip NA
- Neapibrėžtos matematinės reikšmės NaN
- is.na() testuoja ar egzistuoja NA
- is.nan() testuoja ar egzistuoja NaN
- NAN gali turėti klases (integer, numeric)
- NaN yra NA, bet NA nėra NaN

Trūkstami skaičiai

- is.na() ir is.nan() komandos pateikia vektorių, kuriame atspindimas testavimo rezultatas

```
> x <- c(1,2,NA,4,5,6)
> is.na(x)
[1] FALSE FALSE  TRUE FALSE FALSE FALSE
> is.nan(x)
[1] FALSE FALSE FALSE FALSE FALSE FALSE

> x <- c(1,2,NA,4,NaN,6)
> is.na(x)
[1] FALSE FALSE  TRUE FALSE  TRUE FALSE
> is.nan(x)
[1] FALSE FALSE FALSE FALSE  TRUE FALSE
```

Data frames

- Data frames naudojami laikyti tabelinius duomenis
- Iš esmės tai specialus atvejis List, kuriame kiekvienas stulpelis turi būti to paties ilgio
- Kiekvienas stulpelis gali talpinti vis kitos klasės duomenis
- Specialūs data frames atributai
  - rownames
  - colnames
- Dažnai sukuriamos nuskaitant duomenis pvz., read.table() arba read.csv()
- Galima pakeisti į matricą su as.matrix()
- Tuščią *data frame* galima sukurti su data.frame()

Data frames

- Data frames naudojami laikyti tabelinius duomenis

```
> x <-data.frame(FName=c("Ana", "Maria", "John", "Peter"), ↩
    Grades=c(9,10,7,8))
> x
  FName Grades
1   Ana      9
2 Maria     10
3  John      7
4 Peter      8
> nrow(x)
[1] 4
> ncol(x)
[1] 2
> rownames(x)
[1] "1" "2" "3" "4"
> colnames(x)
[1] "FName"  "Grades"
```

Data frames

- Data frames galima priskirti eilučių ir stulpelių pavadinimus

```
> y <- data.frame(1:3)
> y
  X1.3
1    1
2    2
3    3
> colnames(y) <- "NR"
> y
  NR
1  1
2  2
3  3
> rownames(y) <- c("alpha", "beta", "gama")
> y
      NR
alpha  1
beta   2
gama   3
```

List names

- List irgi gali turėti pavadinimus

```
> x <- list (a=1, b=2, c=c(1:3))
> x
$a
[1] 1

$b
[1] 2

$c
[1] 1 2 3
```

Matrix names

- Matricos irgi gali turėti pavadinimus, tik čia tai dimnames()

```
> m <-matrix(1:4, nrow = 2, ncol=2)
> m
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> dimnames(m) <- list(c("a", "b"), c("c", "d"))
> m
  c d
a 1 3
b 2 4
```

Vektorių vardai

- Vektorių įverčiams irgi galima priskirti pavadinimus

```
> x <- 1:3
> x
[1] 1 2 3
> names(x) <- c("a", "b", "c")
> x
a b c
1 2 3
> str(x)
 Named int [1:3] 1 2 3
 - attr(*, "names")= chr [1:3] "a" "b" "c"
```

Masyvai (*arrays*)

- Daugiamačiai masyvai

```
> a <- array(c(1:6),dim = c(3,3,2))
> print(a)
, , 1

     [,1] [,2] [,3]
[1,]    1    4    1
[2,]    2    5    2
[3,]    3    6    3

, , 2

     [,1] [,2] [,3]
[1,]    4    1    4
[2,]    5    2    5
[3,]    6    3    6
```

Duomenų importas į R

Pagrindinės funkcijos, kurios apdeda importuoti duomenis į R

- read.table(), read.csv() tabelinių duomenų importavimui
- readLines nuskaityti tekstą (pvz. .txt, .html)
- source() importuoti R kodo failus
- dget() importuoti R kodo failus
- load() importavimas išsaugotų darbolaukių (workspace)
- unserialize(), importavimas R objektų binarinėje formoje

Duomenų eksportas iš R

Pagrindinės funkcijos, kurios apdeda eksportuoti duomenis į R

- write.table(), write.csv()
- writeLines()
- dump()
- dput()
- save()
- serialize()

read.table()

read.table() pagrindiniai argumentai

- file, nuskaitomo failo pavadinimas
- header, loginis indikatorius, ar egzistuoja stulpelių pavadinimai
- sep, nudoro kaip atskirti stulpeliai
- colClasses, vektorius, nurodantis skirtingas stulpelių klases
- nrows, eilučių skaičius duomenyse
- comment.char, nurodo kaip žymimi komentarai faile
- skip, skaičius, kiek eilučių nuo viršaus praleisti
- stringsAsFactors, ar character variables turėtų būti pakeisti į faktorius
- pilnas sąrašas ?read.table

PVZ: read.table() + CLI + ping

- R'e pasitikriname kur esame: getwd()
- Jeigu reikia, su setwd() pasikeičiame į folderį Sxxx/R/data
- CLI užrašome komandą:

```
ping www.lithuanian-economy.net > c/Users/studentas/Dekstop/↩
    Sxxxx/R/data/LE-ping.txt
```

- leidžiame paveikti surinkti duomenų ir nutraukiam su Ctrl+C
- su Sublime pasižiūrime kaip atrodo duomenys (pas kiekvieną kiek kitaip)
- 1 eilutės nereikia, kaip ir paskutinių...
- pvz:

```
data <-read.table("LE.txt", skip=1, header = FALSE, sep=" ",↩
     nrows = 46, comment.char="")
```

PVZ: read.table() + CLI + ping

- Išdaliname 8 stulpelį į dvi dalis
- Išdalinimas tampa list
- Iš list paverčiame dataframe (apie do.call pakalbėsime vėliau)
- Paverčiame df antrą stulpelį į numeric
- su hist() nupiešiame paprastą histogramą

```
time <-strsplit(as.character(data$V8),'=',fixed=TRUE)
df <-  data.frame(do.call(rbind, time))
df$X2 <- as.numeric(df$X2)
str(df)
hist(df$X2)
```

- PVZ baigtas

Didelių failų nuskaitymas

- Kiek reikia RAM norimiems duomenims:
  - 1 numeric įrašas = 8 bytes
  - 1 000 000 eilučių ir 20 stulpelių
  - $1000000 \times 200 \times 8 = 1880000000$
  - 1024 bytes = KB, 1024 KB = 1 MB, $\frac{bytes}{2^{20}} = MB$, $\frac{bytes}{2^{30}} = GB$
  - $\frac{1880000000}{2^{20}} = 1792.9Mb$ arba 1.75 GB
  - nykščio taisyklė, reikia dvigubai daugiau RAM!
- Naudoti comment.char=""
- Galima padėti R geriau optimizuoti RAM nurodant nrows=...

```
initial <- read.table("data.txt", nrows=100)
classes <- sapply(initial, class)
df <- read.table("data.txt", colClasses=classes,
                                comment.char="")
```

Tekstiniai formatai

- dump(), dput() išsaugo duomenis tekstiniu formatu, bet su meta duomenimis
- dput() skirtas vienam failui
- dump() skirtas vienam arba daugiau failų
- Tekstiniai formatai idealus naudojant VCS
- Tekstiniai formatai yra universalūs, todėl iš esmes atsparūs "zeitgeist"
- Minusas, jog tekstiniai formatai užima daugiau vietos

dump() ir source()

```
AirQualityUCI <- read.csv("AirQualityUCI.csv",
                          sep=";",
                          header = TRUE,
                          stringsAsFactors = FALSE,
                          comment.char = "")
household_power_consumption <- read.table("household_power_↩
    consumption.txt",
                                          sep=";",
                                          header = TRUE,
                                          stringsAsFactors = ↩
                                              FALSE,
                                          comment.char = "")

list_files <-ls()
dump(list_files, file="duomenys.R")
dump(c("AirQualityUCI",
       "household_power_consumption"),
     file="duomenys.R")
rm(list=ls())
source("duomenys.R")
```

dump() ir source()

- dump veikia gerai, kol failai nėra labai dideli (<100mb)
- Reikia patiems įsivertinti, kas yra greičiau dump() + source()
- ar visgi read.table() + visos komandos...

R ir išorinis pasaulis

Kai nuskaitome failą, R naudojasi file funkcija, kad sudarytų ryšį su norimu failu

- file, atidaro ryšį su failu

```
function (description = "", open = "", blocking = TRUE, ↩
    encoding = getOption("encoding"),
    raw = FALSE, method = getOption("url.method", "default")↩
        )
```

- description - failo pavadinimas
- open - "r" (read only), "w" (writing), "a" (appending), "rb", "wb", "ab" (binarinėje formoje)

R ir išorinis pasaulis

- abu variantai tolygūs, nes funkcija read.table viduje naudojasi file funkcija

```
con <- file("LE.txt", "r")
data <- read.table(con, sep=" ",
                   skip = 1,
                   nrows = 156,
                   stringsAsFactors = FALSE,
                   comment.char = "",
                   header = FALSE)
close(con)

data <- read.table("LE.txt",
                   sep=",",
                   skip = 1,
                   nrows = 60,
                   stringsAsFactors = FALSE,
                   comment.char = "")
```

R ir išorinis pasaulis

Galima atidaryti ryšį ir su zipintais failais

- gzfile, atidro ryšį su .gzip failu
- bzfile, atidro ryšį su .bzip2 failu

```
con <- gzfile("census-income.data.gz")
data <- read.table(con, sep=",",
                   nrows = 100,
                   stringsAsFactors = FALSE,
                   comment.char = "",
                   header = FALSE)

close(con)
```

R ir išorinis pasaulis

- url, atidro ryšį su web tinklapiu
- galima nuskaityti pasirinkto tinklapio html kodą

```
con <- url("http://www.delfi.lt", "r")
data <- readLines(con)
close(con)
```

Subsetting

Pagrindiniai operatoriai leidžiantys pasirinkti dalį R objektų

- [...] visada duoda objektą tos pačios klasės, galima pasirinkti daugiau nei vieną elementą
- [[...]] vieno elemento iš list arba dataframe pasirinkimui
- $ leidžia pasirinkti pagal pavadinimus (pagal col.names)

Subsetting

- skaitinis indeksas
- loginis indeksas

```
x <- c("a", "b", "c", "d")

#skaitinis
x[1]
x[2]
x[1:3]

#loginis
x[x>"b"]
rule <- x>"b"
rule
x[rule]
```

Subsetting

- Subsetting naudojant list objektą
- Pradžiai pasidarome šį objektą:

```
> x <- list(grades=1:10,
            names=c("Ana", "Maria", "John", "Peter"),
            course=c("1gr", "2gr"))
```

Subsetting list

```
> x[1]
$grades
 [1]  1  2  3  4  5  6  7  8  9 10

> x[[1]]
 [1]  1  2  3  4  5  6  7  8  9 10

> x$names
[1] "Ana"   "Maria" "John"  "Peter"

> x["names"]
$names
[1] "Ana"   "Maria" "John"  "Peter"

> x[["names"]]
[1] "Ana"   "Maria" "John"  "Peter"
```

Subsetting list

```
> x[c(1,3)]
$grades
 [1]  1  2  3  4  5  6  7  8  9 10

$course
[1] "1gr" "2gr"

> x[[c(1,3)]]
[1] 3
```

Subsetting list

```
> kint <- "names"

> x[kint]
$names
[1] "Ana"   "Maria" "John"  "Peter"

> x[[kint]]
[1] "Ana"   "Maria" "John"  "Peter"

> x$kint
NULL
```

Subsetting list

```
> x <- list(grades=1:10, names=c("Ana", "Maria", "John", "Peter")↩
    ,
+           course=c("1gr", "2gr"))

> x[[2]]
[1] "Ana"    "Maria" "John"   "Peter"

> x[[c(2,2)]]
[1] "Maria"

> x[[2]][[2]]
[1] "Maria"
```

Subsetting

- Subsetting naudojant matricą (i,j)
- Subsetting su [] duoda vektorių, ne matricą!

```
> m <- matrix(1:9, nrow=3, ncol=3)
> m
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

> m[1,1]

[1] 1
> m[3,3]

[1] 9
> m[2,]
[1] 2 5 8

> m[,3]
[1] 7 8 9
```

Subsetting

- Subsetting naudojant matricą (i,j)
- Subsetting su [] duoda vektorių, ne matricą, todėl drop=FALSE

```
> m <- matrix(1:9, nrow=3, ncol=3)

> m[1,1, drop=FALSE]
     [,1]
[1,]    1

> m[2,, drop=FALSE]
     [,1] [,2] [,3]
[1,]    2    5    8

> m[,3, drop=FALSE]
     [,1]
[1,]    7
[2,]    8
[3,]    9
```

NA išvalymas

- Kartais duomenyse yra NA

```
> x <- c(1,2,3,NA,5,6,NA,8)
> y <-c("a", "b", "c", NA, NA, "f", "g" , "h")

> is.na(x)
[1] FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE

> trukst_vek <- is.na(x)

> x[trukst_vek]
[1] NA NA

> x[!trukst_vek]
[1] 1 2 3 5 6 8
```

NA išvalymas

- complete.cases() duoda loginį vektorių, su pozicijomis, kuriose nėra NA

```
> complete.cases(x,y)
[1]  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE

> x[complete.cases(x,y)]
[1] 1 2 3 6 8

> y[complete.cases(x,y)]
[1] "a" "b" "c" "f" "h"
```

NA išvalymas

- Su complete.cases() galima išvalyti ir dataframe

```
> library(datasets)
> airquality[1:6,]
  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

> airquality[complete.cases(airquality),][1:6,]
  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
7    23     299  8.6   65     5   7
8    19      99 13.8   59     5   8
```

NA išvalymas

- kita alternatyva: na.omit()

```
> library(datasets)
> airquality[1:6,]
  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

> na.omit(airquality)[1:6,]
  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
7    23     299  8.6   65     5   7
8    19      99 13.8   59     5   8
```

NA išvalymas

O bet tačiau...

- O bet tačiau... ypatingai atliekant apklausas, visada nutiks taip, jog dalis respondentų neatsakys į kuriuos nors pavienius klausimus (pvz., nesupras klausimo)
- Visos obzervacijos panaikinimas gali būti labai "brangus", ypač turint mažą n
- Todėl geriau atliekant skaičiavimus su R, funkcijoms nurodyti kaip apeiti NA
- Alternatyva, pakeisti NA pvz 0, arba vidutine kintamojo reikšme. Tačiau tai būtina protokoluoti ir nurodyti tyrime / tyrimo meta apraše

Vektorizuotos operacijos

- R skaičiavimus atlieka vektorizuojant savo objektus

```
> x<- 1:5; y<-3:7; z<- 1:2

> x+y
[1]  4  6  8 10 12
> x*y
[1]  3  8 15 24 35
> x/y
[1] 0.3333333 0.5000000 0.6000000 0.6666667 0.7142857

> x+z
[1] 2 4 4 6 6
Warning message:
In x + z : longer object length is not a multiple of shorter ↩
    object length

> x*z
[1] 1 4 3 8 5
Warning message:
In x * z : longer object length is not a multiple of shorter ↩
    object length
```

Matricos

```
> x<-matrix(1:4,2,2); y<-matrix(rep(10,4),2,2); s<-matrix(1:2,↩
    nrow=2)

> x
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> y
     [,1] [,2]
[1,]   10   10
[2,]   10   10
> x*y
     [,1] [,2]
[1,]   10   30
[2,]   20   40
> x+y
     [,1] [,2]
[1,]   11   13
[2,]   12   14
```

Matricos

```
> x<-matrix(1:4,2,2); y<-matrix(rep(10,4),2,2); s<-matrix(1:2,↩
    nrow=2)

> x%*%y
     [,1] [,2]
[1,]   40   40
[2,]   60   60
> x%*%s
     [,1]
[1,]    7
[2,]   10
```

Valdymo struktūros (control structures) leidžia valdyti programų veikimą, priklausomai nuo tam tikrų aplinkybių:

- if, else: testuoja tam tikrą aplinkybę
- for: vykdo programą tam tikrą iteracijų skaičių
- while: vykdo programą kol egzistuoja tam tikros aplinkybės
- repeat: vykdo nesibaigiančią iteraciją
- break: nutraukia iteracijos procesą
- next: peršoka 1 iteraciją
- return: nutraukia funkciją

if

```
# 1
if(<condition>) {
        ## do something
}

#2
if(<condition>) {
        ## do something
} else{
        ## do something
}

#3
if(<condition>) {
        ## do something
} else if(<condition>) {
        ## do something
} else{
        ## do something
}
```

if

```
x <- 5

if(x>3){
        y <- 5
} else{
        y <- 0
}

# tapatu

y <- if (x>3) {
        5
} else{
        0
}
```

for

- for loop dažniausiai naudojami iteruoti tam tikriems veiksmams, žinant, kiek kartų iteracija turi trukti.
- galima naudoti "i" arba bet kokią kitą raidę / stringą

```
for (i in 1:5){
        print(i)
}

for (values in 1:5){
        print(values)
}

[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

for

- visi šie for loops veikia vienodai

```
x <- c("a", "b", "c", "d", "e", "f")

for (i in 1:6) {
        print(x[i])
}

for (i in seq_along(x)){
        print(x[i])
}

for (raide in x) {
        print(raide)
}

for(i in 1:6) print(x[i])
```

for

- nested for loop
- retai naudojama, sunkiai suprantama, geriau nekišt nagų

```
x <- matrix(1:9, ncol=3, nrow=3)

for (i in seq_len(nrow(x))) {
        for (j in seq_len(ncol(x))) {
                print(x[i,j])
        }

}

seq_len()
```

while

- while testuoja aplinkybes, jeigu ok, atlieka veiksmą, pabaigus vėl testuoja ir t.t.
- while loop gali testis neribotą skaičių iteracijų, tad atsargiai
- galima sutikti, kai bandoma iteruoti optimizavimo uždavinius, kur while (<condition>) yra siekiama vertė

```
count <-0

while(count<5000){
        print(count)
        count <-count+1
}
```

while

- pvz., čia visai neaišku, kada baigsis iteracija

```
z<- 5

while(z>=3 && z<=10){
        print(z)
        #rbinom(n, size, prob)
        coin <- rbinom(1,1,0.5)
        if(coin==1){
                z<- z+1
        }else{
                z<- z-1
        }
}
```

repeat, break, next

- repeat inicializuoja begalinės trukmės loop
- vienintelis būdas sustabdyti, su break
- pavojinga funkcija!

```
x0 <- 1
tol <- 1e-8

repeat{
        x1 <- computeEstimate() #pvz kokia nors optimizavimo ↩
            funkcija
        if(abs(x1-x0)<tol){
                break
        }else{
                x0 <-x1
        }
}
```

repeat, break, next

- next komanda peršoka prie sekančios iteracijos

```
for (i in 1:100) {
        if(i<=30){
                ## skips the first 30
                next
        }
        print(i)

}
```

R Funkcijos

- Užrašome savo pirmą funkciją
- Ką ji daro?

```
add2 <- function(x,y){
        x+y
}

add2(3,5)
```

R Funkcijos

- 2 funkcija
- Ką ji daro?

```
above10 <- function(x){
        use <- x>10
        x[use]
}

c <- seq(1:20)

above10(c)
```

R Funkcijos

- Patobuliname antrą funkciją: 2 argumentai ir antras standartizuotas argumentas

```
above <- function(x,y){
        use <- x>y
        x[use]
}
above(c,6)

above <- function(x,y=10){
        use <- x>y
        x[use]
}

above(c)
above(c,2)
```

R Funkcijos

- Funkcija, kuri apskaičiuoja stulpelių vidurkius:

```
column_mean <- function(y){
        nc <- ncol(y)
        means <- numeric(nc)
        for (i in 1:nc){
                means[i] <-mean(y[,i])
        }
        means
}

library(datasets)
column_mean(airquality)
```

R Funkcijos

- Kai kurių stulepelių nepavyko apskaičiuoti, nes kai kuriuos reikšmės NA
- na.rm=TRUE funkcijoje mean(), leidžia apskaičiuoti įverčius pašalinant NA

```
column_mean <- function(y){
        nc <- ncol(y)
        means <- numeric(nc)
        for (i in 1:nc){
                means[i] <-mean(y[,i], na.rm = TRUE )
        }
        means
}
```

R Funkcijos apibendrinimas

- Funkcijos savaime yra R objektai, kuriuos galima perduoti kaip argumentus kitoms funkcijoms
- Funkcijos gali būti *nested* viena į kitą
- Funkcijos rezultatas - paskutinė R išraiška funkcijos viduje
- *Formal arguments* - predefinuoti argumentai, tai palengvina funkcijų naudojimą (pvz., read.csv: sep=",")

```
f <- function (<arguments>){
        ##  funkcijos veikla
}
```

R Funkcijos

- Pozicinis vs leksinis, dalinis funkcijų argumentų *matching*
  - Leksikinis pilnas matching
  - Leksikinis dalinis, bet unikalus matching
  - Pozicinis matching

```
?sd
sd(x, na.rm = FALSE)

data <- rnorm(100)

sd(data)
sd(x=data)
sd(x=data, na.rm = TRUE)
sd(na.rm = TRUE, x=data)
sd(na.rm = TRUE, data)
```

R Funkcijos

- Pozicinis vs leksinis, dalinis funkcijų argumentų *matching*
  - Leksikinis pilnas matching
  - Leksikinis dalinis, bet unikalus matching
  - Pozicinis matching
- Patarimas bent jau pradžioje išrašykite argumentus su jų apibrėžimu, padeda greičiau išmokti ir padarysite mažiau klaidų

```
args(lm)
function (formula, data, subset, weights, na.action, method = "qr↩
    ", model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok↩
     = TRUE, contrasts = NULL, offset, ...)

mydata <- data.frame(x=rnorm(1000), y=rnorm(1000))

lm(data=mydata, y~x, model = FALSE, 1:100)
lm(y~x, data=mydata, 1:100, model = FALSE)
lm(y~x, dat=mydata, 1:100, mod = FALSE)
lm (formula= "y~x", data=mydata, subset=1:100,model = FALSE)
plot(lm(formula= "y~x", data=mydata, subset=1:100,model = FALSE))
```

R Funkcijos - Lazy evaluation

- *Lazy evaluation* reiškia, jog argumentai funkcijoje panaudojami tada, kai ir jeigu, jų reikia

```
f<-function(a,b){
          a^2
}
f(2) #positional matching a=2
[1] 4

f <- function(a,b){
        print(a)
        print(b)
}

f(10)  #positional matching a=10

[1] 10
Error in print(b) : argument "b" is missing, with no default
```

R Funkcijos - ...

- ... indikuoja argmentus, kurie perduodami kitai funkcijai
- *Generic functions* naudoja ... *methods* (šiuo metu nesvarbu...)

```
myplot <- function(x,y,type="l",...){
        plot(x,y,type=type,...)
}

plot(mydata$x, mydata$y)
myplot(mydata$x, mydata$y)
```

R Funkcijos - ...

- ... indikuoja argmentus, kurie perduodami kitai funkcijai
- *Generic functions* naudoja ... *methods* (šiuo metu nesvarbu...)

```
myplot <- function(x,y,type="l",...){
        plot(x,y,type=type,...)
}

plot(x=rnorm(1000), y=rnorm(1000))
myplot(x=rnorm(1000), y=rnorm(1000))
```

R Funkcijos - ...

- ... arba kai funkcija negali žinoti, kokie argumentai bus pateikti, arba kiek jų
- Tačiau po jų, būtina teisingai išrašyti argumentus

```
args(paste)
function (..., sep = " ", collapse = NULL)

args(cat)
function (..., file = "", sep = " ", fill = FALSE, labels = NULL,
          append = FALSE)
paste("a", "b", "c", sep = ",")
[1] "a,b,c"

paste("a", "b", "c", se = ",")
[1] "a b c ,"
```

Data ir laikas

- Datos turi Date klasę
- Laikas gali būti POSIXct arba POSIXlt klasės
- Data išsaugoma kaip dienų skirtumas lyginant su 1970-01-01
- Laikas išsaugomas kaip sekundžių skirtumas lyginant su 1970-01-01

Data ir laikas

- Character string su data galima paversti į datos klasės objektą

```
x <- as.Date("2019-03-27")

x
[1] "2019-03-27"

class(x)
[1] "Date"

unclass(x)
[1] 17982
```

Data ir laikas

- Laikas gali b8ti POSIXct arba POSIXlt klasės
- POSIXct išsaugo laiką kaip skaičių
- POSIXlt išsaugo laiką kaip *list* su daug papildomos informacijos
- Naudingos funkcijos
  - weekdays
  - months
  - quarters

```
x <- Sys.time()
x
class(x)
p <- as.POSIXlt(x)
p
unclass(p)
names(unclass(p))
p$sec

unclass(x)
```

Data ir laikas

- Komanda strptime padeda iš *charackter string* nuskaityti datą
- Praktikoje patartina geriau naudotis paketai tokiais kaip *lubridate*, *zoo*

```
datestring <- c("2019 January 21, 21:15","2019 February 14, ↩
    14:14")
x <- strptime(datestring, format="%Y %B %d, %H:%M")
x
class(x)

#but not with lithuanian names
datestring <- c("2019 Sausis 21, 21:15","2019 Vasaris 14, ↩
    14:14")
x <- strptime(datestring, format="%Y %B %d, %H:%M")
x
```

Data ir laikas

- Su datomis galima pasižaisti:

```
x <- as.Date("2019-03-27")
class(x)

y <- strptime("2019 January 21, 21:15", format="%Y %B %d, %H:%M")

x-y

as.POSIXct(x)-y
as.POSIXlt(x)-y
```

Data ir laikas

- Su datomis galima pasižaisti:

```
x <- as.Date("2016-02-28"); y <- as.Date("2016-03-01")
y-x
x <- as.Date("2016-02-28"); y <- as.Date("2016-02-29")
y-x

lt <- as.POSIXct("2019-02-27 08:00:00", tz = "EET")
us <- as.POSIXct("2019-02-27 08:00:00", tz = "EST")
us-lt
```

Tvarkingas kodavimas

- Visada rašykite kodą su (plain text) editoriumi
- Naudokite *identing* (8 spaces) (CRTL+I @R)
- Max eilučių ilgis: 80 ženklų
- Apribokite funkcijas: 1 funkcija - 1 operacija