Duomenų analizės įvadas

2.1. dalis - R programavimas

Justas Mundeikis

VU EVAF

2019-03-06

Turinys

Apie R ## 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)

R Istorija

- R veikia 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
- Paleidžiame R
- Tam kad būtų lengviau dirbti su R, turėti aibę papildomų funkcijų, instaliuojame Rstudio
- Startuojame RStudio

R sistema

- R susideda iš dviejų komponentų:
  - Bazinė R sistema su standartiniais paketais
  - Visų kitų paketų

```
search()
##  [1] ".GlobalEnv"         "package:kableExtra" "package:stats"
##  [4] "package:graphics"   "package:grDevices"  "package:utils"
##  [7] "package:datasets"   "package:methods"    "Autoloads"
## [10] "package:base"
```

- 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

```
list_packages <- available.packages()
length(list_packages)
## [1] 230367
```

Kur rasti pagalbą

Visi susidursite su problemomis, kai kažkas neveiks kaip norite, kai R praneš apie klaidas 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`
3. Kursiokai / Mokslo grupė
4. stackoverflow.com
5. Dėstytojas (žr. sekanti skaidrė)

Stackoverflow

Programuotojų bendruomenė (kartais grubi ir nelabai supratinga), todėl problemą reikia aprašyti trumpai ir aiškiai:

- Antraštė turėtų būti trumpa ir aiški
- 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

R Input Output

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"
```

- Kai neužbaigta įvestis, R rodo +, tada arba pabaigt ivestį arba ESC

R rezervuoti objektai

- `?reserved` komanda parodo, kokie objektų pavadinimai negali būti perrašyti

```
?reserved
```

- nebent naudojamos backticks kabutes pvz.: `` `if ` ``

```
`if` <- ...
```

R loginiai operatoriai

```
v <- c(3,1,TRUE,2+3i,0,0); t <- c(4,1,FALSE,2+3i,1,0)
```

& 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 FALSE FALSE
```

&& is 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
```

R loginiai operatoriai

| 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  TRUE  TRUE  TRUE  TRUE FALSE
```

|| is 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] TRUE
```

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

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

R Input Output

Operatorius ":" sukuria eiles (sequence). Tačiau yra ir komanda seq()

```
x <- 1:5
x
## [1] 1 2 3 4 5
#tapatu
z <- c(1,2,3,4,5)
z
## [1] 1 2 3 4 5
# daugiau galimybių su atitinkama funkcija
y <- seq(from=5, to=10, by=0.5)
y
##  [1]  5.0  5.5  6.0  6.5  7.0  7.5  8.0  8.5  9.0  9.5 10.0
```

Objektų tipai

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

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
x
## [1] 2.1
Inf
## [1] Inf
1/Inf
## [1] 0
0/0
## [1] NaN
```

R duomenų tipai

- Vektoriai
- Matricos
- Data frame
- List
- Arrays

R duomenų atributai

R objektai gali turėti atributus:

- names, dimnames
- dimensions (e.g matricos 2x2 2x3 3x2 ir t.t.)
- class (numeric, charackter)
- length (`x <- vector(length=5)`)
- kiti vartotojo priskirti atributai
- `attributes()` leidžia nustatyti / keisti objekto atributus

Vektorių sukūrimas

- c() funkcija leidžia sukurti objektų vektorius
- c() iš ***c***oncatenate

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

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

Vektorių sukūrimas

- Vektorius galima rankiniu būdu priskirti tam tikrai klasei

```
x<-0:5
class(x)
## [1] "integer"
as.numeric(x)
## [1] 0 1 2 3 4 5
as.logical(x)
## [1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
as.character(x)
## [1] "0" "1" "2" "3" "4" "5"
as.factor(x)
## [1] 0 1 2 3 4 5
## Levels: 0 1 2 3 4 5
x <-c(0,0,1,1,2,2,3,3)
as.factor(x)
## [1] 0 0 1 1 2 2 3 3
## Levels: 0 1 2 3
```

Vektorių sukūrimas

- tačiau nelogiški priskyrimai generuos NAs

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

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"
```

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

- Jau egzistuojančiam vektoriui galima suteikti dimensijas post factum

```
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
```

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
```

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)
## Warning in cbind(x, y): number of rows of result is not a multiple of
## vector length (arg 1)
##      x y
## [1,] 1 1
## [2,] 2 2
## [3,] 3 3
## [4,] 1 4
## [5,] 2 5
rbind(x,y)
## Warning in rbind(x, y): number of columns of result is not a multiple of
## vector length (arg 1)
##   [,1] [,2] [,3] [,4] [,5]
## x    1    2    3    1    2
## y    1    2    3    4    5
```

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
```

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"
rownames(y) <- c("alpha", "beta", "gama")
y
##       NR
## alpha  1
## beta   2
## gama   3
```

List - sąrašas

- [[*nr*]] nurodo list objekto numerį

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

List names

- List irgi gali turėti pavadinimus

```
x <- list (a=1, b=(1:3), c=c("alpha", "beta" , "gamma"))
x
## $a
## [1] 1
##
## $b
## [1] 1 2 3
##
## $c
## [1] "alpha" "beta"  "gamma"
```

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 priskiriami 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
```

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

Duomenų importas į R

Pagrindinės funkcijos, kurios padeda 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 binarine forma

Duomenų eksportas iš R

Pagrindinės funkcijos, kurios padeda eksportuoti duomenis į R

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

read.table()

Funkcijos `read.table()` pagrindiniai argumentai (`?read.table`)

- `file`: nuskaitomo failo pavadinimas turi būti nurodytas su kabutėmis pvz., `faile="data.csv"`
- `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 (patrainta visada `=FALSE`)

Praktinis intarpas

Šio praktinio interpo tiklsas: išmatuoti tinklapio veikimo / atsakymo greitį. Šis veiksmas atliekmas CLI su komanda `ping`. Surinktus duomenis importuosime į R ir apdorojus nubraižysime tinklapio reakcijos greičio histogramą.

- Pisitkrinama kur yra CLI `pwd`.
  - jeigu ne norimame darbiniame folderyje (pvz., "175" esančiame ant desktopo), tada `cd /Desktop`
  - jeigu nėra darbinio folderio (pvz., "175"), tada `mkdir S175`
- `cd S175`

Praktinis intarpas

- Ping komanda su nukreipimu į tekstinį failą:

```
ping -flag http://... > ping-data.txt
```

- Linux:
  - `ping` kol nebus nutraukta su Ctrl+C
  - `ping -c 200` kol surinks 200 pingų
- Windows:
  - `ping -t` kol nebus nutraukta su Ctrl+C
  - `ping -n 200` kol surinks 200 pingų

Praktinis intarpas

Sekantis žingsnis, su editoriumi pvz., *Sublime* išsinalaizuoti gautą failą:

- ar yra nereikalingų eilučių, kurias reiktų praleisti nuo viršaus (`skip=`)?
- ar yra antraštės (`header=`)?
- ar yra eilučių apačioje, kurių nenorim imprtuoti (`nrow=`)?
- kaip yra atskirti stulpeliai (`sep=`)?
- ar duomenys turi tam tikrą regioninį kodavimą (`fileEncoding =`)

Pastaba, jeigu nepavyko, failas yra mano Github paskyroje “ping-test.txt” (su 500 pingų)

Praktinis intarpas

ping-test.txt atveju:

- pirmos ir antros eilutės nereikia ("PING www.lithuanian-economy.net (178.254.62.91) 56(84) bytes of data."), todėl `skip=2`
- antraščių nėra, todėl `header=FALSE`
- stulepliai atskirti SPACE, todėl `sep=" "`
- pabaigoje yra eilučių kurių nereikia. paskutinė man aktuali eilutė - 500 (502 - 2 skippintos eilutės), todėl `nrow=500`

Praktinis intarpas

- pasitikriname kur yra R darbinė direktorija `getwd()` (CLI buvo pwd)
- jeigu direktorija bloga, keičiame su `setwd()` (CLI buvo cd)
- pvz., `setwd("/c/Users/studentas/Desktop/S175")` (reikalingos kabutės!) veikia TAB, tad padeda supildyti be klaidų
- esant teisingoje dirketorijoje, galima pasitikrinti, ar egzistuoja norimas failas su `dir()` (CLI buvo -ls)

```
df <-  read.table("./duomenys_paskaitoms/ping-data.txt",
                  skip = 2,
                  header = FALSE,
                  sep="",
                  nrow=500,
                  stringsAsFactors = FALSE)
```

Praktinis intarpas

- Pasižiūrime į pirmas 3 eilutes:

```
head(df,3)
##      V1   V2              V3       V4         V5      V6
## 1 Reply from 178.254.62.91: bytes=32 time=33ms TTL=127
## 2 Reply from 178.254.62.91: bytes=32 time=33ms TTL=127
## 3 Reply from 178.254.62.91: bytes=32 time=33ms TTL=127
```

- ir į paskutines 3 eilutes:

```
tail(df, 3)
##        V1   V2              V3       V4         V5      V6
## 498 Reply from 178.254.62.91: bytes=32 time=32ms TTL=127
## 499 Reply from 178.254.62.91: bytes=32 time=33ms TTL=127
## 500 Reply from 178.254.62.91: bytes=32 time=34ms TTL=127
```

Praktinis intarpas

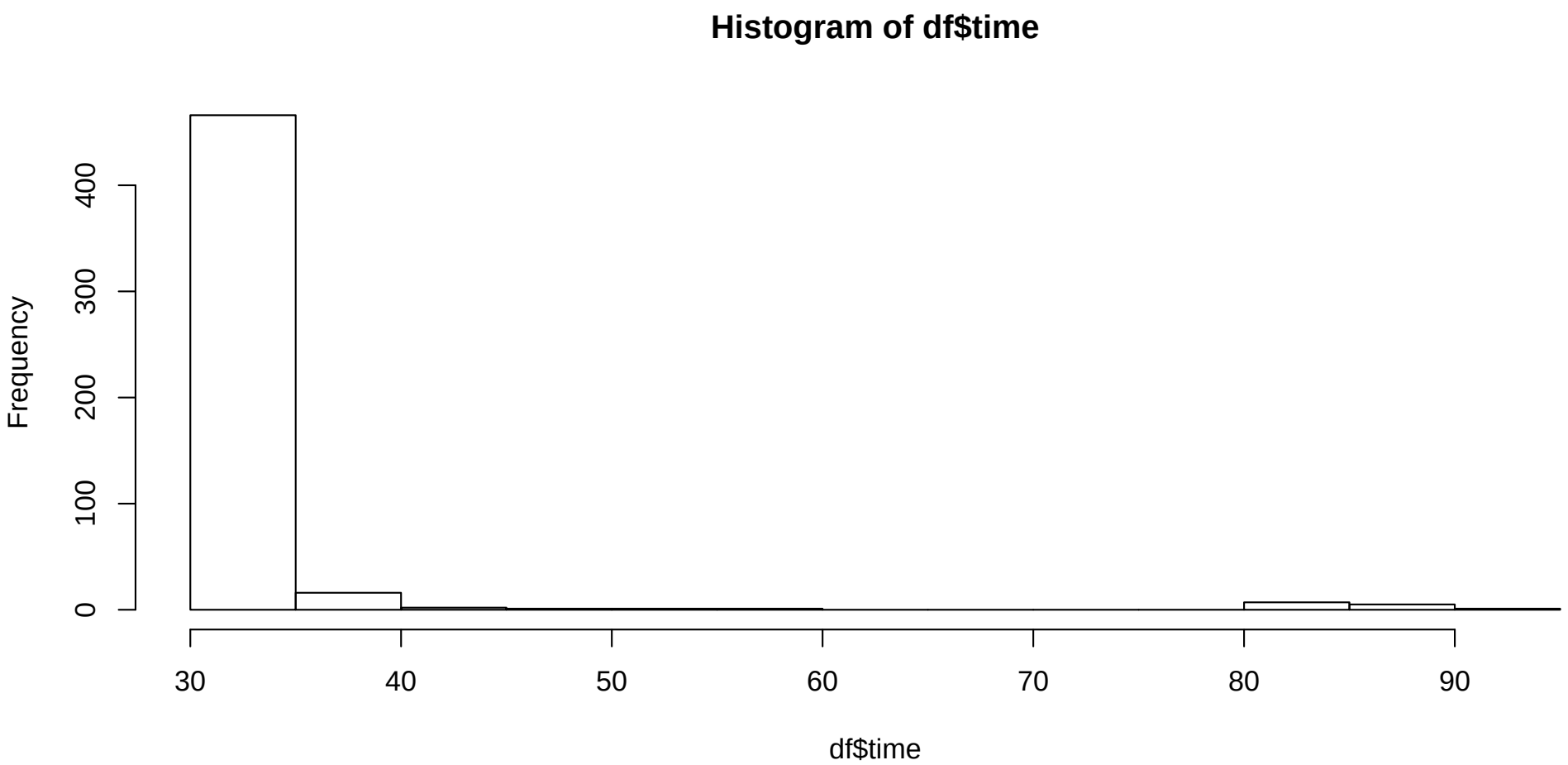
- Windows ir Linux išsaugomi ping duomenys gali skirtis:
- Linux'e laikas patalipintas į 8 stulepelį (laikas ir "ms" atskriti)
- Windows'e į 5 stulpelį
- Iš šio stulpelio reikia išsitraukti 6 ir 7 ženklą (pagal eiliškumą), patalpinti gautus duomenis į naują stulpelį "time", kuris prisegamas "df" pabaigoje (dešinėje)
- Jeigu pinginamas tolimas serveris ir laikas gali susidėti iš 3 ženklų, atitinkamai koreguoti į 6 ir 8 ženklą

```
df$time <- as.numeric(substr(df[,5], 6,7))
```

Praktinis intarpas

Belieka tik nubraižyti histogramą

```
hist(df$time)
```



Praktinis intarpas

Beje, labai patogu su *Sublime* manualiai koreguoti kad ir daug eilučių turinčius failus:

- pirma manualiai ištriname nereikalingas eilutes viršuje ir apačioje
- CTRL+A užmarkiruoja visą tekstą
- CTRL+SHIFT+L įjungia editavimo modusą visų pasirinktų eilučių pabaigoje

Tekstiniai formatai

- `dump()`, `dput()` išsaugo duomenis tekstiniu formatu kartu 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()` 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

R ir išorinis pasaulis

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

- file, atidaro ryšį su failu

```
file
## function (description = "", open = "", blocking = TRUE, encoding = getOp
##     raw = FALSE, method = getOption("url.method", "default"))
## {
##     .Internal(file(description, open, blocking, encoding, method,
##         raw))
## }
## <bytecode: 0x55dc759ff8f0>
## <environment: namespace:base>
```

- 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("./duomenys_paskaitoms/ping-data.txt", "r")
df <- read.table(con,
                 sep=" ",
                 skip = 2,
                 nrows = 500,
                 stringsAsFactors = FALSE,
                 comment.char = "",
                 header = FALSE)
close(con)
df <- read.table("./duomenys_paskaitoms/ping-data.txt",
                 sep=" ",
                 skip = 2,
                 nrows = 500,
                 stringsAsFactors = FALSE,
                 comment.char = "",
                 header = FALSE)
```

R ir išorinis pasaulis

Galima atidaryti ryšį ir su zipintais failais

- gzfile, atidro ryšį su .gzip failu
- bzfile, atidro ryšį su .bzip2 failu
- problematiška, jeigu zip faile ne tik nuskaitomi duomenys bet ir pvz., meta aprašas

```
con <- gzfile("./duomenys_paskaitoms/census-income.data.gz")
# https://archive.ics.uci.edu/ml/datasets/Census-Income+%28KDD%29
census_data <- read.table(con,
                          sep=",",
                          nrows = 100,
                          stringsAsFactors = FALSE,
                          comment.char = "",
                          header = FALSE)
```

R ir išorinis pasaulis

- url, atidro ryšį su web tinklapiu
- galima nuskaityti pasirinkto tinklapio html kodą
- arba duomenis, kurie yra atviri

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

# atitinka:
delfi_html <- readLines("http://delfi.lt")
```

Subsetting

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 galimas naudajant:
  - skaitinį indeksą
  - loginį indeksą

Subsetting

Skaitinis indeksas

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

#skaitinis
x[1]
## [1] "a"
x[2]
## [1] "b"
x[1:3]
## [1] "a" "b" "c"
```

Subsetting

Loginis indeksas

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

#loginis
x[x>"b"]
## [1] "c" "d"
rule <- x>"b"
rule
## [1] FALSE FALSE  TRUE  TRUE
x[rule]
## [1] "c" "d"
```

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
x[is.na(x)] <- 0
x
## [1] 1 2 3 0 5 6 0 8
```

NA išvalymas

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

```
complete.cases(x,y)
## [1]  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE
x[complete.cases(x,y)]
## [1] 1 2 3 6 0 8
y[complete.cases(x,y)]
## [1] "a" "b" "c" "f" "g" "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 nedidelį respondentų skaičių
- 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
## Warning in x + z: longer object length is not a multiple of shorter obje
## length
## [1] 2 4 4 6 6
x*z
## Warning in x * z: longer object length is not a multiple of shorter obje
## length
## [1] 1 4 3 8 5
```

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
```

R programavimas

Valdymo struktūros

Valdymo struktūros

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 kartotis
- galima naudoti “i” arba bet kokią kitą raidę / stringą

```
for (i in 1:4){
        print(i)
}
## [1] 1
## [1] 2
## [1] 3
## [1] 4

for (values in 1:4){
        print(values)
}
## [1] 1
## [1] 2
## [1] 3
## [1] 4
```

for

- visi šie for loops veikia vienodai

```
x <- c("a", "b", "c", "d")
for (i in 1:4) {
        print(x[i])
}
## [1] "a"
## [1] "b"
## [1] "c"
## [1] "d"

for (i in seq_along(x)){
        print(x[i])
}
## [1] "a"
## [1] "b"
## [1] "c"
## [1] "d"
```

for

- visi šie for loops veikia vienodai

```
for (raide in x) {
        print(raide)
}
## [1] "a"
## [1] "b"
## [1] "c"
## [1] "d"

for(i in 1:4) print(x[i])
## [1] "a"
## [1] "b"
## [1] "c"
## [1] "d"
```

for

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

```
x <- matrix(1:6, ncol=3, nrow=2)

for (i in seq_len(nrow(x))) {
        for (j in seq_len(ncol(x))) {
                print(x[i,j])
        }
}
## [1] 1
## [1] 3
## [1] 5
## [1] 2
## [1] 4
## [1] 6
#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 () yra siekiama vertė

```
count <-0

while(count<5){
        print(count)
        count <-count+1
}
## [1] 0
## [1] 1
## [1] 2
## [1] 3
## [1] 4
```

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
        }
}
## [1] 5
## [1] 4
## [1] 5
## [1] 6
## [1] 5
## [1] 4
```

repeat, break, next

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

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

repeat{
        x1 <- computeEstimate() #puz 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)
        }
}
```

Funkcijos

R Funkcijos

- Užrašome savo pirmą funkciją
- Ką ji daro?

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

add2(3,5)
## [1] 8
```

R Funkcijos

- 2 funkcija
- Ką ji daro?

```
above10 <- function(x){
        use <- x>10
        x[use]
}
c <- seq(1:20)
above10(c)
##  [1] 11 12 13 14 15 16 17 18 19 20
```

R Funkcijos

- Patobuliname antrą funkciją: 2 argumentai ir antras standartizuotas argumentas

```
above <- function(x,y){
        use <- x>y
        x[use]
}
above(c,6)
##  [1]  7  8  9 10 11 12 13 14 15 16 17 18 19 20
above <- function(x,y=10){
        use <- x>y
        x[use]
}
above(c)
##  [1] 11 12 13 14 15 16 17 18 19 20
above(c,2)
##  [1]  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

R Funkcijos

- Funkcija, kuri apskaičiuoja stulpelių vidurkius:

```
column_mean <- function(y){
        nc <- ncol(y)
        means <- numeric(length = nc) #creates a numeric vector
        for (i in 1:nc){
                means[i] <-mean(y[,i])
        }
        print(means)
}

library(datasets)
column_mean(airquality)
## [1]        NA        NA  9.957516 77.882353  6.993464 15.803922
```

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 )
        }
        print(means)
}
column_mean(airquality)
## [1]  42.129310 185.931507   9.957516  77.882353   6.993464  15.803922
```

R Funkcijos apibendrinimas

- Funkcijos savaime yra R objektai, kuriuos galima perduoti kaip argumentus kitoms funkcijoms (*first class object*)
- Funkcijos gali būti *nested* viena į kitą
- Funkcijos rezultatas - paskutinė R išraiška funkcijos viduje, todel jeigu reikia `print()` kaip paskutinį funkcijos veiksmą
- *Formal arguments* - predefinuoti argumentai, tai palengvina funkcijų naudojima (pvz., `read.csv sep=","`)

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

R Funkcijos *matching*

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

```
# sd(x, na.rm = FALSE)
data <- rnorm(100)

sd(data)
## [1] 0.9856431
sd(x=data)
## [1] 0.9856431
sd(x=data, na.rm = TRUE)
## [1] 0.9856431
sd(na.rm = TRUE, x=data)
## [1] 0.9856431
sd(na.rm = TRUE, data)
## [1] 0.9856431
```

R Funkcijos *matching*

- 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, ...)
## NULL
```

R Funkcijos

PVZ:

```
mydata <- data.frame(x=rnorm(200), y=rnorm(200))

# plot demonastracija rankiniu būdų
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

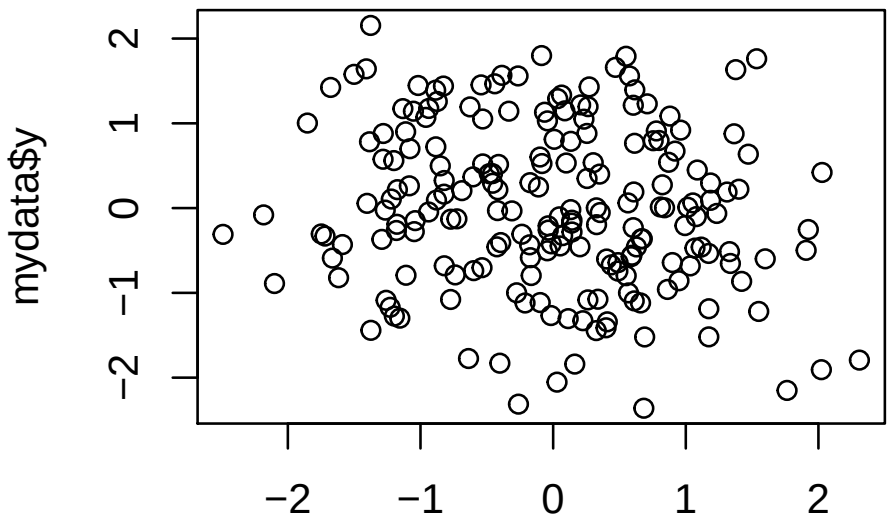
- *Lazy evaluation* reiškia, jog argumentai funkcijoje panaudojami tada, kai ir jeigu, jų reikia

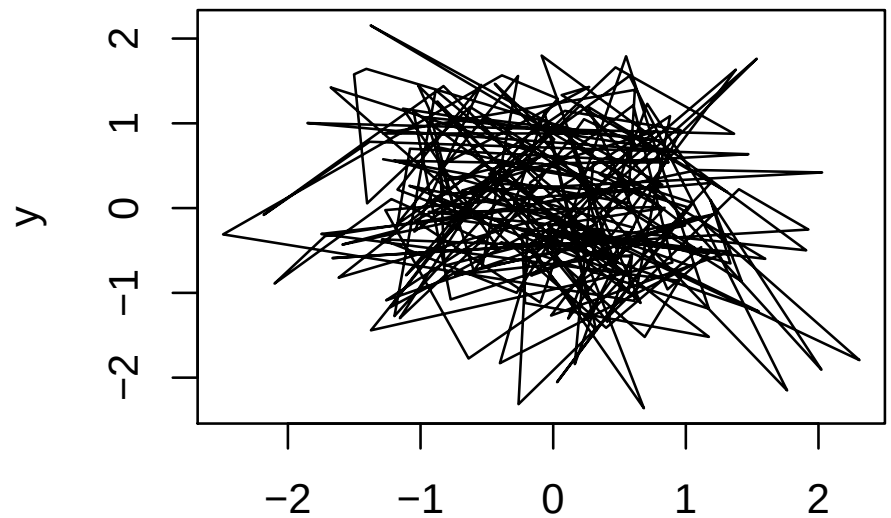
```
f<-function(a,b){
        a^2
}
f(2) # positional matching a=2
## [1] 4

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...)

```
mydata <- data.frame(x=rnorm(200), y=rnorm(200))
myplot <- function(x,y,type="l",...){
        plot(x,y,type=type,...)
}
par(mfrow=c(1,2))
plot(mydata$x, mydata$y)
myplot(mydata$x, mydata$y)
```





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)
## NULL
args(cat)
## function (..., file = "", sep = " ", fill = FALSE, labels = NULL,
##     append = FALSE)
## NULL

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

Data ir laikas

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")
print(x)
## [1] "2019-03-27"
class(x)
## [1] "Date"
unclass(x)
## [1] 17982
```

Data ir laikas

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

Data ir laikas

```
x <- Sys.time()
print(x)
## [1] "2019-03-06 13:24:56 EET"
class(x)
## [1] "POSIXct" "POSIXt"
p <- as.POSIXlt(x)
class(p)
## [1] "POSIXlt" "POSIXt"
unclass(p)
## $sec
## [1] 56.49252
##
## $min
## [1] 24
##
## $hour
## [1] 13
##
## $mday
## [1] 6
```

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")
print(x)
## [1] "2019-01-21 21:15:00 EET" "2019-02-14 14:14:00 EET"
class(x)
## [1] "POSIXlt" "POSIXt"

# bet ne su lietuviškais pavadinimais
datestring <- c("2019 Sausis 21, 21:15","2019 Vasaris 14, 14:14")
x <- strptime(datestring, format="%Y %B %d, %H:%M")
print(x)
## [1] NA NA
```

Data ir laikas

- Su datomis galima pasižaisti:

```
x <- as.Date("2019-03-27")
class(x)
## [1] "Date"

y <- strptime("2019 January 21, 21:15", format="%Y %B %d, %H:%M")
class(y)
## [1] "POSIXlt" "POSIXt"
x-y
## Warning: Incompatible methods ("-.Date", "-.POSIXt") for "-"
## Error in x - y: non-numeric argument to binary operator

as.POSIXct(x)-y
## Time difference of 64.19792 days
as.POSIXlt(x)-y
## Time difference of 64.19792 days
```

Data ir laikas

- POSIX atpažįsta laiko zonas ir t.t.

```
x <- as.Date("2016-02-28"); y <- as.Date("2016-03-01")
y-x
## Time difference of 2 days
x <- as.Date("2016-02-28"); y <- as.Date("2016-02-29")
y-x
## Time difference of 1 days

lt <- as.POSIXct("2019-02-27 08:00:00", tz = "EET")
us <- as.POSIXct("2019-02-27 08:00:00", tz = "EST")
us-lt
## Time difference of 7 hours
```

Tvarkingas kodavimas

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