

Duomenų analizės įvadas

2-1 Dalis

Justas Mundeikis

2019 m. vasario 26 d.

1. Dalies turinys

R Istorija

- R yra S dialektas
- S parašyta John Chambers et al. @Bell Labs 1976m.
- S buvo perrašyta 1988 (v3) ir tapo labiau panaši į statistinę programą, o 1998 išleista S v4.
- R sukurtas 1991 mokslinio darbo rėmuose (Ross Ihaka ir Robert Gentleman)
- 1993 R pristatytas visuomenei
- 1995 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
- **Philosophy of the GNU Project**

Free software

Jeigu kalbame apie "free software" turima omenyje

- Laisvė naudotis programa, bet kuriuo tikslu
- Laisvė analizuoti ir keisti programą pagal savo poreikius
- Laisvė dalintis programos kopijomis
- Laisvė dalintis pagerintomis kopijomis

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

- Bazinė R sistema su standartiniais paketais: stats, graphics, grDevices, utils, datasets, methods, base
- Visi kiti paketai
- dauguma R paketų saugomi CRAN (Comprehensive R Archive Network), iš kur atsisieniuiamas ir pats R
- `available.packages()` funkcija, kuri surenką visą informaciją apie egzistuojančius R paketus @CRAN

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

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

Kur rasti pagalbą

- Dokumentacija
- Google (Ctrl+C Ctrl+V error code)
- Kursiškai
- Stackoverflow
- Dėstytojas

Stackoverflow

- <https://stackoverflow.com/>
- Kokius konkrečiai žingsnius atlikote
- Kokio rezultato tikėtės
- 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, ">" prompt (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 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 (klasės)

- character
- numeric
- integer
- complex
- logical

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

- 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 sukūrimas

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

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

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

Vektorių sukūrimas

- vektorius galima priskirti tam tikrai klasei manualiai

```
> 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, tas pats vektoriaus objektas, tačiau turintis dimensiją

```
> 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 stulpelinio būdu, 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 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 coercion'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
- NA gali turėti klases (integer, character)
- NaN yra NA, bet NA nėra NaN

Trūkstami skaičiai

- Pvz

```
> 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()`
- tuščia 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"
```

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, nurodo kaip atskirti stulpeliai
- colClasses, vektorius, nurodantis skirtingas stulpelių klases
- nrow, 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, $\frac{\text{bytes}}{2^{20}} = \text{MB}$, $\frac{\text{bytes}}{2^{30}} = \text{GB}$
 - $\frac{1880000000}{2^{20}} = 1792.9 \text{ Mb}$ 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 esmės atsparūs "zeitgeist"
- Minusai, 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 patiemis įsivertinti, kas yra greičiau dump() + source()
- ar visgi read.table() + visos komandos...

R ir išorinis pasaulis

Kai nuskaitytume failą, R naudojami 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, atidaro ryšį su .gzip failu
- bzfile, atidaro 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"

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

Subsetting list

- Subsetting naudojant list objektą

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

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

> x$kint
NULL
```

Subsetting list

- Subsetting naudojant list objektą

```
> 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

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 observacijos 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
```