

## Predavanja 3 – data.frame, indeksiranje

### Indeksiranje vektorjev

V praksi nas pogosto ne zanima celoten vektor ali celotna tabela, ampak samo določeni elementi, npr. operacijo želimo izvesti samo na nekem stolpcu ali izbrati samo vrstice, ki ustrezajo nekemu pogoju. Do posameznih elementov največkrat dostopamo preko t. i. indeksiranja – navajanja elementov, ki jih želimo izbrati. Pogledali si bomo dva načina indeksiranja:

1. Podamo vektor z zaporednimi števkami elementov, ki jih želimo izbrati.
2. Podamo vektor enake dolžine kot elementi, ki jih izbiramo, pri čemer vsak element tega vektorja pove, ali istoležni element izberemo ali ne.

Najprej ustvarimo nov vektor:

```
vek <- c(4, 2, 3, 6, 7, 1)
```

Kako dostopamo do posameznih elementov oziroma do podmnožic tega vektorja? Lahko uporabimo prvi način, se pravi z zaporednimi števkami elementov, ki jih želimo izbrati.

Izberemo četrty elementa vektorja.

```
vek[4]
```

```
## [1] 6
```

Izberemo od drugega do petega elementa vektorja.

```
vek[2:5]
```

```
## [1] 2 3 6 7
```

Vse, kar smo se naučili pri vektorjih, pride tudi v poštev pri indeksiranju, npr. `c()` ali `seq()`

```
vek[c(2, 4, 6)]
```

```
## [1] 2 6 1
```

```
vek[seq(2, 6, by = 2)]
```

```
## [1] 2 6 1
```

Vrstni red elementov lahko tudi obrnemo:

```
vek[seq(6, 2, by=-2)]
```

```
## [1] 1 6 2
```

Pri drugem načinu potrebujemo nov vektor, enake dolžine kot **vek**, kjer bodo istoležeči elementi povedali, katere elemente v **vek** želimo izbrati. Pri tem se srečamo s posebnim tipom spremenljivke, ki se ji reče **logična spremenljivka** ali **boolean**. Elementi tega vektorja lahko zavzamejo samo vrednosti **TRUE** in **FALSE**, ali skrajšano **T** in **F**. Izbrani bodo tisti elementi vektorja **vek**, kjer bodo vrednosti tega novega vektorja enake **TRUE**.

Izberemo drugi, četrti in šesti element vektorja na drugi način, z uporabo logičnega vektorja.

```
lv <- c(F, T, F, T, F, T)
vek[lv]
```

```
## [1] 2 6 1
```

Izbrani so tisti elementi vektorja **vek**, kjer ima istoležni element vektorja **lv** vrednosti **T**.

Navajanje **T** in **F** za vsak element vektorja je v praksi zelo nepraktično, saj imamo običajno opravka z vektorji dolžine reda velikosti 100, 1000 ali več. V tem primeru je navajanje številskih indeksov veliko bolj praktično. Prednost tega pristopa se pokaže, ko **T/F** ne navajamo ročno, ampak gre za rezultate neke logične operacije (pogoja). Na primer, ustvarimo logičen vektor na podlagi pogoja:

```
po <- vek > 3
print(po)
```

```
## [1] TRUE FALSE FALSE TRUE TRUE FALSE
```

Ta vektor lahko uporabimo kot argument vektorja **vek**.

```
po <- vek > 3
vek[po]
```

```
## [1] 4 6 7
```

```
vek[vek > 3]
```

```
## [1] 4 6 7
```

Poznamo več logičnih operatorjev:

- **>** Je večje.
- **<** Je manjše.
- **==** Je enako. Potrebno je biti pozoren, da potrebujemo dva enačaja, saj je en enačaj rezerviran za prirejanje vrednosti!
- **>=** Je večje ali enako.
- **<=** Je manjše ali enako.
- **|** Ali.
- **&** In.

- ! Negiranje.
- != Ni enako.

Vsi standardni matematični in logični operatorji so nepogrešljivi pri programiranju.

Poglejmo si nekaj primerov logičnih operatorjev.

```
x <- c(1, 2, 5, 6, 3, 2, 2, 1)
x == 2
```

```
## [1] FALSE TRUE FALSE FALSE FALSE TRUE TRUE FALSE
```

```
x >= 3
```

```
## [1] FALSE FALSE TRUE TRUE TRUE FALSE FALSE FALSE
```

```
x == 2 | x == 3
```

```
## [1] FALSE TRUE FALSE FALSE TRUE TRUE TRUE FALSE
```

```
x > 1 & x < 6
```

```
## [1] FALSE TRUE TRUE FALSE TRUE TRUE TRUE FALSE
```

```
x != 5
```

```
## [1] TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE
```

Če želimo izbrati na primer vse elemente, kjer je  $x$  večji od 1 in manjši od 6 lahko uporabimo:

```
x[x > 1 & x < 6]
```

```
## [1] 2 5 3 2 2
```

Poglejmo si še posebno funkcijo `which()`, ki pretvori logični vektor v indekse elementov, ki imajo vrednost T.

```
which(x == 2)
```

```
## [1] 2 6 7
```

## data.frame

V R-ju je **data.frame** dvodimenzionalna podatkovna struktura, sestavljena z vrstic in stolpcev. Poznamo sicer tudi druge dvodimenzionalne strukture (**matrix**, **table**, **tibble**), ampak `data.frame` je najpogostejše uporabljena. `data.frame` si lahko predstavljamo kot nekakšno Excel tabelo shranjeno v R.

Kako naredimo `data.frame`? Naredimo več vektorjev iste dolžine in jih združimo.

```
visina <- c(179, 185, 183, 172, 174, 185, 193, 169, 173, 168)
teza <- c(75, 89, 70, 80, 58, 86, 73, 63, 72, 70)
spol <- c("f", "m", "m", "m", "f", "m", "f", "f", "m", "f")
df <- data.frame(spol, visina, teza)
print(df)
```

```
##      spol visina teza
## 1     f    179   75
## 2     m    185   89
## 3     m    183   70
## 4     m    172   80
## 5     f    174   58
## 6     m    185   86
## 7     f    193   73
## 8     f    169   63
## 9     m    173   72
## 10    f    168   70
```

Vsak vektor predstavlja en stolpec data.frame-a. Vektorji morajo biti iste dolžine. Če vektorji niso iste dolžine, R vrne napako:

```
visina <- c(179, 185, 183, 172, 174, 185, 193, 169, 173, 168)
teza <- c(75, 89, 70, 80, 58, 86, 73, 63)
spol <- c("f", "m", "m", "m", "f", "m", "f", "f", "m")
df <- data.frame(spol, visina, teza)
```

```
## Error in data.frame(spol, visina, teza): arguments imply differing number of rows: 9, 10, 8
```

Podatke lahko tudi preberemo iz datoteke:

```
dat <- read.table("./data_raw/president_county_candidate.csv",
                  header=TRUE,
                  sep=";",
                  quote = "")
```

Tabela je velika in je ne moremo izpisati v celoti.

Če želimo pogledati samo nekaj začetnih vrstic:

```
head(dat)
```

```
##      state      county      candidate party  votes
## 1 Delaware Kent County Joe Biden    DEM  44518
## 2 Delaware Kent County Donald Trump  REP  40976
## 3 Delaware Kent County Jo Jorgensen  LIB   1044
## 4 Delaware Kent County Howie Hawkins GRN    420
## 5 Delaware Kent County Write-ins    WRI     0
## 6 Delaware New Castle County Joe Biden  DEM 194238
```

Že želimo pogledati 20 začetnih vrstic:

```
head(dat, 20)
```

##	state	county	candidate	party	votes
## 1	Delaware	Kent County	Joe Biden	DEM	44518
## 2	Delaware	Kent County	Donald Trump	REP	40976
## 3	Delaware	Kent County	Jo Jorgensen	LIB	1044
## 4	Delaware	Kent County	Howie Hawkins	GRN	420
## 5	Delaware	Kent County	Write-ins	WRI	0
## 6	Delaware	New Castle County	Joe Biden	DEM	194238
## 7	Delaware	New Castle County	Donald Trump	REP	87685
## 8	Delaware	New Castle County	Jo Jorgensen	LIB	2932
## 9	Delaware	New Castle County	Howie Hawkins	GRN	1278
## 10	Delaware	New Castle County	Write-ins	WRI	0
## 11	Delaware	Sussex County	Donald Trump	REP	71196
## 12	Delaware	Sussex County	Joe Biden	DEM	56657
## 13	Delaware	Sussex County	Jo Jorgensen	LIB	1003
## 14	Delaware	Sussex County	Howie Hawkins	GRN	437
## 15	District of Columbia	District of Columbia	Joe Biden	DEM	29509
## 16	District of Columbia	District of Columbia	Donald Trump	REP	1149
## 17	District of Columbia	District of Columbia	Write-ins	WRI	186
## 18	District of Columbia	District of Columbia	Howie Hawkins	GRN	185
## 19	District of Columbia	District of Columbia	Jo Jorgensen	LIB	134
## 20	District of Columbia	District of Columbia	Gloria La Riva	PSL	73

Če želimo pogledati zadnjih nekaj vrstic:

```
tail(dat)
```

##	state	county	candidate	party	votes
## 31300	Arizona	Maricopa County	Joe Biden	DEM	944285
## 31301	Arizona	Maricopa County	Donald Trump	REP	880347
## 31302	Arizona	Maricopa County	Jo Jorgensen	LIB	25747
## 31303	Arizona	Mohave County	Donald Trump	REP	74553
## 31304	Arizona	Mohave County	Joe Biden	DEM	23993
## 31305	Arizona	Mohave County	Jo Jorgensen	LIB	1189

Če želimo pogledati zadnjih 20 vrstic:

```
tail(dat, 20)
```

##	state	county	candidate	party	votes
## 31286	Arizona	Coconino County	Donald Trump	REP	26212
## 31287	Arizona	Coconino County	Jo Jorgensen	LIB	1471
## 31288	Arizona	Gila County	Donald Trump	REP	18241
## 31289	Arizona	Gila County	Joe Biden	DEM	8875
## 31290	Arizona	Gila County	Jo Jorgensen	LIB	340
## 31291	Arizona	Graham County	Donald Trump	REP	10747
## 31292	Arizona	Graham County	Joe Biden	DEM	4034
## 31293	Arizona	Graham County	Jo Jorgensen	LIB	212
## 31294	Arizona	Greenlee County	Donald Trump	REP	2433
## 31295	Arizona	Greenlee County	Joe Biden	DEM	1182

```
## 31296 Arizona Greenlee County Jo Jorgensen LIB 70
## 31297 Arizona La Paz County Donald Trump REP 4542
## 31298 Arizona La Paz County Joe Biden DEM 2050
## 31299 Arizona La Paz County Jo Jorgensen LIB 79
## 31300 Arizona Maricopa County Joe Biden DEM 944285
## 31301 Arizona Maricopa County Donald Trump REP 880347
## 31302 Arizona Maricopa County Jo Jorgensen LIB 25747
## 31303 Arizona Mohave County Donald Trump REP 74553
## 31304 Arizona Mohave County Joe Biden DEM 23993
## 31305 Arizona Mohave County Jo Jorgensen LIB 1189
```

Če nas zanima velikost data.frame-a:

```
dim(dat)
```

```
## [1] 31305 5
```

Če nas zanima koliko vrstic ima data.frame:

```
nrow(dat)
```

```
## [1] 31305
```

Če nas zanima koliko stolpcev ima data.frame:

```
ncol(dat)
```

```
## [1] 5
```

Imena stolpcev data.frame-a:

```
names(dat)
```

```
## [1] "state" "county" "candidate" "party" "votes"
```

Osnovno statistiko posameznih stolpcev v data.frame-u dobimo z ukazom `summary`:

```
summary(dat)
```

```
##      state      county      candidate      party
## Length:31305 Length:31305 Length:31305 Length:31305
## Class :character Class :character Class :character Class :character
## Mode :character Mode :character Mode :character Mode :character
##
##
##      votes
## Min.   : 0
## 1st Qu.: 2
## Median :32
## Mean   :4704
## 3rd Qu.:762
## Max.   :2486527
```

data.frame ni samo 2D tabela, ampak nosi tudi podatke o tem, kakšen tip spremenljivke je posamezen stolpec. Vsak stolpec je lahko drug tip spremenljivke, vsi elementi v stolpcu morajo biti istega tipa (kot pri vektorjih). Opazimo, da so vsi razen zadnjega stolpca besedilni (v R-ju **character**), zadnji pa je številski (**numeric**).

## Indeksiranje data.frame

Včasih želimo dostopati samo do posameznih stolpcev ali vrstic data.frame-a (ali do neke podmnožice vrstic in/ali stolpcev). Zaradi lažjega razumevanja bomo kot primer uporabili kar majhen, na roke ustvarjen data.frame **df** (glej zgoraj). S spodaj navedenimi ukazi samo izpišemo izbrane elemente, v **df** so še vedno shranjeni vsi elementi. Kako izbrisati vrstice ali stolpce iz spremenljivke **df** ali pa shraniti podmnožico tega data.frame, bomo spoznali kasneje.

Izberemo tretjo vrstico

```
df[3, ]
```

```
##   spol visina teza
## 3    m    183   70
```

Izberemo od tretje do šeste vrstice:

```
df[3:6, ]
```

```
##   spol visina teza
## 3    m    183   70
## 4    m    172   80
## 5    f    174   58
## 6    m    185   86
```

Izberemo tretji stolpec:

```
df[, 3]
```

```
## [1] 75 89 70 80 58 86 73 63 72 70
```

Izberemo prvi do drugi stolpec:

```
df[, 1:2]
```

```
##   spol visina
## 1    f    179
## 2    m    185
## 3    m    183
## 4    m    172
## 5    f    174
## 6    m    185
## 7    f    193
## 8    f    169
## 9    m    173
## 10   f    168
```

Izberemo prvi in tretji stolpec:

```
df[, c(1, 3)]
```

```
##      spol teza
## 1      f   75
## 2      m   89
## 3      m   70
## 4      m   80
## 5      f   58
## 6      m   86
## 7      f   73
## 8      f   63
## 9      m   72
## 10     f   70
```

Izberemo prvi in tretji stolpec in drugo, četrto in šesto vrstico:

```
df[c(2, 4, 6), c(1, 3)]
```

```
##      spol teza
## 2      m   89
## 4      m   80
## 6      m   86
```

Stolpce lahko “pokličemo” kar z njihovim imenom:

```
df[, "spol"]
```

```
## [1] "f" "m" "m" "m" "f" "m" "f" "f" "m" "f"
```

```
df[, c("spol", "teza")]
```

```
##      spol teza
## 1      f   75
## 2      m   89
## 3      m   70
## 4      m   80
## 5      f   58
## 6      m   86
## 7      f   73
## 8      f   63
## 9      m   72
## 10     f   70
```

Posamezen stolpec v R-ju lahko izberemo tudi z operatorjem \$:

```
df$spol
```

```
## [1] "f" "m" "m" "m" "f" "m" "f" "f" "m" "f"
```



To nam pomaga pri iskanju podatkov, ki nas zanimajo. Naredimo podobno kot smo naredili pri vektorjih. Želimo samo osebe višje od 180 cm. Uporabili bomo logično indeksiranje:

```
df[df$visina > 180, ]
```

```
##   spol visina teza
## 2    m   185   89
## 3    m   183   70
## 6    m   185   86
## 7    f   193   73
```

Želimo samo osebe, ki so višje od 175 cm in so ženske:

```
df[df$visina > 175 & df$spol == "f", ]
```

```
##   spol visina teza
## 1    f   179   75
## 7    f   193   73
```

Za zgoraj izbrano skupino žensk želim vedeti koliko so težke:

```
df[df$visina > 175 & df$spol == "f", "teza"]
```

```
## [1] 75 73
```

Lahko tudi:

```
df[df$visina > 175 & df$spol == "f", ]$teza
```

```
## [1] 75 73
```

Pazljivi moramo biti, da med pogojem za vrstice in stolpce napišemo vejico, tudi če pogoja za stolpce ni. To vrne **Error**.

```
df[df$visina > 180]
```

```
## Error in '[.data.frame'(df, df$visina > 180): undefined columns selected
```

## Odstranjevanje in dodajanje vrstic in stolpcev

V R-ju lahko **odstranimo** stolpec kar z operatorjem `-`. Uporabimo izraz **odstranimo**, ampak v bistvu s spodaj opisanimi postopki samo prikažemo del `df` brez določenih vrstic ali stolpcev (`df` ostane kakršen je bil). S temi izrazi izberemo vse stolpce/vrstice, razen tistih, ki so navedeni za operatorjem `-`.

Želimo odstraniti drugi stolpec:

```
df[, -2]
```

```
##      spol teza
## 1      f   75
## 2      m   89
## 3      m   70
## 4      m   80
## 5      f   58
## 6      m   86
## 7      f   73
## 8      f   63
## 9      m   72
## 10     f   70
```

Odstranimo tretjo vrstico:

```
df[-3, ]
```

```
##      spol visina teza
## 1      f    179   75
## 2      m    185   89
## 4      m    172   80
## 5      f    174   58
## 6      m    185   86
## 7      f    193   73
## 8      f    169   63
## 9      m    173   72
## 10     f    168   70
```

Odstranimo od tretje do šeste vrstice:

```
df[-(3:6), ]
```

```
##      spol visina teza
## 1      f    179   75
## 2      m    185   89
## 7      f    193   73
## 8      f    169   63
## 9      m    173   72
## 10     f    168   70
```

Odstranimo od tretje do šeste vrstice in prvi do drugi stolpec:

```
df[-(3:6), -(1:2)]
```

```
## [1] 75 89 73 63 72 70
```

Odstranimo drugo, četrto in šesto vrstico in prvi stolpec:

```
df[-c(2,4,6), -1]
```

```
##      visina teza
## 1      179   75
## 3      183   70
## 5      174   58
## 7      193   73
## 8      169   63
## 9      173   72
## 10     168   70
```

Ne deluje pa z imeni:

```
df[, -c("spol", "teza")]
```

```
## Error in -c("spol", "teza"): invalid argument to unary operator
```

Stolpec dodamo tako, da najprej ustvarimo vektor, ki je take dolžine, kolikor ima data.frame vrstic:

```
dim(df)
```

```
## [1] 10  3
```

```
imena <- c("Micka", "Marko", "Gregor", "Tomaz", "Ana", "Peter",
           "Mojca", "Katja", "Anze", "Alja")
print(imena)
```

```
## [1] "Micka" "Marko" "Gregor" "Tomaz" "Ana"   "Peter" "Mojca" "Katja"
## [9] "Anze"  "Alja"
```

```
length(imena)
```

```
## [1] 10
```

```
df$imena <- imena
print(df)
```

```
##      spol visina teza  imena
## 1      f   179   75  Micka
## 2      m   185   89  Marko
## 3      m   183   70 Gregor
## 4      m   172   80  Tomaz
## 5      f   174   58   Ana
## 6      m   185   86  Peter
## 7      f   193   73  Mojca
## 8      f   169   63  Katja
## 9      m   173   72   Anze
## 10     f   168   70   Alja
```

Vrstico ustvarimo tako, da najprej ustvarimo data.frame, ki ima iste attribute kot `df` in vrstico dodamo na konec `df`:

```
vrstica <- data.frame(spol = "m", visina = 170, teza = 60, imena = 'Samo')
dim(df)
```

```
## [1] 10 4
```

```
df[11, ] <- vrstica
print(df)
```

```
##      spol visina teza  imena
## 1      f    179   75  Micka
## 2      m    185   89  Marko
## 3      m    183   70 Gregor
## 4      m    172   80  Tomaz
## 5      f    174   58   Ana
## 6      m    185   86  Peter
## 7      f    193   73  Mojca
## 8      f    169   63  Katja
## 9      m    173   72   Anze
## 10     f    168   70   Alja
## 11     m    170   60   Samo
```

Oziroma če želimo, da dela vedno, ne samo, ko dodajamo enajsto vrstico:

```
df[nrow(df) + 1, ] <- vrstica
print(df)
```

```
##      spol visina teza  imena
## 1      f    179   75  Micka
## 2      m    185   89  Marko
## 3      m    183   70 Gregor
## 4      m    172   80  Tomaz
## 5      f    174   58   Ana
## 6      m    185   86  Peter
## 7      f    193   73  Mojca
## 8      f    169   63  Katja
## 9      m    173   72   Anze
## 10     f    168   70   Alja
## 11     m    170   60   Samo
## 12     m    170   60   Samo
```

Sedaj smo na konec še enkrat dodali Samo. Kako ga odstranimo?

```
df <- df[-nrow(df), ]
print(df)
```

```
##      spol visina teza  imena
## 1      f    179   75  Micka
## 2      m    185   89  Marko
## 3      m    183   70 Gregor
## 4      m    172   80  Tomaz
```

```
## 5      f      174    58    Ana
## 6      m      185    86   Peter
## 7      f      193    73   Mojca
## 8      f      169    63   Katja
## 9      m      173    72    Anze
## 10     f      168    70    Alja
## 11     m      170    60    Samo
```

Z zgornjo kodo smo spremenili `df` (`df <- df[-nrow(df),]`). Torej v `df` smo prepisali `df` brez zadnje vrstice.

## Domača naloga

1. Preberite podatke v mapi `data_raw` o ameriških volitvah. Podatke smo pobrali 6. novembra iz: [https://www.kaggle.com/unanimad/us-election-2020?select=president\\_county\\_candidate.csv](https://www.kaggle.com/unanimad/us-election-2020?select=president_county_candidate.csv). Podatke preberite v `data.frame` in preverite:

- Izberite stolpec **candidate**.

```
## [1] "Joe Biden"      "Donald Trump"  "Jo Jorgensen"  "Howie Hawkins"
## [5] "Write-ins"     "Joe Biden"
```

Prikazanih je prvih 6 vnosov.

- Izberite vrstico številka 500, kaj predstavlja?

```
##      state      county  candidate party votes
## 500 Florida St. Lucie County Donald Trump  REP 86748
```

- Kaj piše v stolpcu **party** in vrstici 645?

```
## [1] "REP"
```

- Odstranite zadnjih 100 vrstic.

```
##      state      county  candidate party votes
## 31300 Arizona Maricopa County  Joe Biden  DEM 944285
## 31301 Arizona Maricopa County Donald Trump  REP 880347
## 31302 Arizona Maricopa County Jo Jorgensen  LIB 25747
## 31303 Arizona Mohave County Donald Trump  REP 74553
## 31304 Arizona Mohave County  Joe Biden  DEM 23993
## 31305 Arizona Mohave County Jo Jorgensen  LIB 1189
```

Izpisanih je zadnjih 6 vrstic.

- Odstranite podatke o okraju (**county**).

```
##      state      candidate party votes
## 1 Delaware  Joe Biden  DEM 44518
## 2 Delaware Donald Trump  REP 40976
## 3 Delaware Jo Jorgensen  LIB 1044
## 4 Delaware Howie Hawkins  GRN 420
## 5 Delaware Write-ins  WRI 0
## 6 Delaware  Joe Biden  DEM 194238
```

Izpisanih je prvih 6 vrstic.

- Izberite podatke samo za zvezno državo Georgia.

```
##      state      county  candidate party votes
## 592 Georgia Appling County Donald Trump  REP  6526
## 593 Georgia Appling County   Joe Biden  DEM  1779
## 594 Georgia Appling County Jo Jorgensen  LIB   36
## 595 Georgia Atkinson County Donald Trump  REP  2300
## 596 Georgia Atkinson County   Joe Biden  DEM   825
## 597 Georgia Atkinson County Jo Jorgensen  LIB   30
```

Izpisanih je prvih 6 vrstic.

- Izberite podatke za zvezno državo Georgia in okraj Clarke County.

```
##      state      county  candidate party votes
## 697 Georgia Clarke County   Joe Biden  DEM 35882
## 698 Georgia Clarke County Donald Trump  REP 14385
## 699 Georgia Clarke County Jo Jorgensen  LIB   831
## 700 Georgia Clarke County  Write-ins  WRI    0
```

- Izberite vse vrstice, kjer je nekdo dobil več kot 100000 glasov.

```
##      state      county  candidate party votes
## 6  Delaware New Castle County   Joe Biden  DEM 194238
## 103 Florida Brevard County Donald Trump  REP 207700
## 104 Florida Brevard County   Joe Biden  DEM 148403
## 110 Florida Broward County   Joe Biden  DEM 617689
## 111 Florida Broward County Donald Trump  REP 332960
## 150 Florida Collier County Donald Trump  REP 128820
```

Izpisanih je prvih 6 vrstic.

- (Težje) Izberi vstico, kjer so Libertariani dobili največ glasov. Namig: vrstico **party** == "LIB", kjer je **votes** največji izmed vseh vrstic **party** == "LIB".

```
##      state      county  candidate party votes
## 28476 California Los Angeles County Jo Jorgensen  LIB 25953
```

2. Preberite podatke v mapi *data\_raw* o BDP na prebivalca evropskih držav *EU\_GDP\_per\_capita.csv*, ki smo jih pobrali 29. septembra 2021 iz spletne strani Eurostat (<https://ec.europa.eu/eurostat/web/national-accounts/data/database>).

```
## country y2000 y2001 y2002 y2003 y2004 y2005 y2006 y2007 y2008 y2009 y2010
## 1      AT 31710 31990 32360 32520 33200 33710 34700 35870 36280 34830 35390
## 2      BE 29890 30110 30490 30680 31640 32200 32800 33760 33640 32700 33330
## 3      BG 3010 3230 3440 3640 3890 4190 4500 4820 5140 4990 5050
## 4      CH 52080 52640 52260 51860 52970 54180 56000 57720 58620 56750 58170
## 5      CY 20130 20700 21230 21520 22300 23050 23740 24420 24680 23550 23400
## 6      CZ 11230 11620 11830 12260 12840 13650 14530 15250 15500 14690 15020
```

```
## y2011 y2012 y2013 y2014 y2015 y2016 y2017 y2018 y2019 y2020
## 1 36300 36390 36180 36130 36140 36390 37030 37800 38170 35610
## 2 33460 33490 33490 33870 34360 34620 35050 35510 35950 33560
## 3 5300 5350 5400 5530 5790 6050 6310 6550 6840 6600
## 4 58730 58820 59200 59920 60220 60780 61170 62500 62800 60820
## 5 22900 21780 20400 20250 21020 22270 23200 24120 24530 23050
## 6 15310 15170 15160 15480 16290 16670 17490 17990 18460 17340
```

- Izberite vrstico, ki prikazuje BDP na prebivalca za Francijo (koda FR).

```
## country y2000 y2001 y2002 y2003 y2004 y2005 y2006 y2007 y2008 y2009 y2010
## 16 FR 28930 29290 29410 29440 30050 30320 30850 31400 31310 30250 30690
## y2011 y2012 y2013 y2014 y2015 y2016 y2017 y2018 y2019 y2020
## 16 31210 31160 31170 31320 31540 31770 32360 32820 33320 30610
```

- Prikažite BDP na prebivalca za Hrvaško v letih 2015, 2017, 2018, 2019 in 2020.

```
## y2015 y2017 y2018 y2019 y2020
## 17 10630 11600 12040 12450 11500
```

- Prikažite razliko BDP-ja na prebivalca med Avstrijo (AT) in Italijo (IT) v zadnjih treh letih.

```
## y2018 y2019 y2020
## 1 10770 10960 10720
```

- Prikažite imena držav, ki so imele leta 2005 BDP na prebivalca nižjega od 10000 ali višjega od 40000 evrov.

```
## [1] "BG" "CH" "DK" "HU"
```

- Preberite tabelo o BDP na prebivalca za Slovenijo in njene vrednosti dodajte v tabelo ostalih EU držav.

```
## country y2000 y2001 y2002 y2003 y2004 y2005 y2006 y2007 y2008 y2009 y2010
## 17 HR 8190 8420 8900 9400 9780 10200 10700 11250 11460 10640 10520
## 18 HU 7910 8250 8670 9050 9510 9930 10340 10390 10510 9820 9960
## 19 IE 33270 34490 35890 36370 38170 39460 40360 41240 38550 36240 36700
## 20 IS 28570 29310 29220 29670 31620 33190 34310 36360 36230 33480 32650
## 21 IT 27430 27950 27960 27850 28040 28100 28490 28740 28250 26600 26940
## 22 SI 11076 11714 12504 13156 13878 14551 15676 17373 18757 17758 17749
## y2011 y2012 y2013 y2014 y2015 y2016 y2017 y2018 y2019 y2020
## 17 10530 10310 10300 10310 10630 11100 11600 12040 12450 11500
## 18 10180 10090 10310 10770 11210 11480 12010 12680 13270 12640
## 19 36940 36770 37060 40010 49620 50060 53930 58100 60130 62980
## 20 33150 33330 34510 34710 35870 37610 38280 39090 39160 36030
## 21 27030 26160 25620 25620 25860 26240 26730 27030 27210 24890
## 22 18052 17626 17700 18253 18830 19589 20820 22136 23167 22312
```