Predavanje 06 – Programski tok, if, for

Cilji predavanja

- Pogojni stavki (if, else).
- For zanka.
- Funkcija `aggregate`.

Motivacijski primer

V mapi *data_raw* se nahajajo podatki o spletnih nakupih darilnih in zabavnih izdelkov v Excelovih datotekah. Podatki so bili del znanstvene raziskave Chen, Sain, and Guo (2012), dostopni so na https://archive.ics.uci.edu/ml/datasets/Online+Retail. V mapi sta dve datoteki s temi podatki in sicer *online-retail.xlsx*, ki ste jo spoznali že na drugih predavanjih in vsebuje 3 liste/države, ter *online-retail-large.xlsx*, ki vsebuje 37 listov/držav. Naš cilj je prebrati vse liste in jih združiti v enoten `data.frame` v R. Nato bomo podatke agregirali glede na državo in izrisali stolpični diagram, ki bo prikazoval, v kateri državi ima podjetje največ prihodka. Za to analizo bomo potrebovali vse 3 koncepte iz Ciljev predavanja.

Programski tok

Skripte, ki jih napišemo, izvajajo ukaze zaporedno iz vrha navzdol. Včasih želimo nekatere ukaze preskočiti, druge pa ponoviti večkrat. To lahko storimo ročno, če uporabljamo bližnjico **Ctrl** + **Enter**, če pa uporabimo ukaz `source()` se skripta izvede v celoti. Če želimo, da se nekateri deli skripte avtomatsko preskočijo oziroma ponovijo, potrebujemo nekaj ukazov, ki nam bodo to omogočili. Primere, kjer bi to potrebovali, bomo spoznali v nadaljevanju.

Ukaz if

Ukaz **if** nam omogoča, da se izbrani ukazi izvedejo samo, če velja določen pogoj. V R-ju to napišemo takole.

```
if (<pogoj>) {
  # Ukazi, ki se poženejo, če velja pogoj.
}
```

V zgornjem primeru je `<pogoj>` lahko spremenljivka ali izraz, katerega rezultat je **TRUE** ali **FALSE**, torej tipa **boolean**. Ukazi znotraj zavitih oklepajev **{}** se izvedejo le v primeru, če je pogoj **TRUE**.

Odprimo najprej prvi list krajše datoteke *online-retail.xlsx*.

```
library(openxlsx)
```

```
## Warning: package 'openxlsx' was built under R version 4.0.4
```

```
podatki <- read.xlsx("./data_raw/online-retail.xlsx", 1)
head(podatki)
```

```
##   InvoiceNo StockCode                         Description Quantity UnitPrice
## 1   C538971     22153     ANGEL DECORATION STARS ON DRESS      -48      0.42
## 2    539330     37449  CERAMIC CAKE STAND + HANGING CAKES        8      8.50
## 3    539330     37446  MINI CAKE STAND WITH HANGING CAKES        8      1.45
## 4    539330     22962               JAM JAR WITH PINK LID       12      0.85
## 5    539330     21428 SET3 BOOK BOX GREEN GINGHAM FLOWER         4      4.25
## 6    539330     22113         GREY HEART HOT WATER BOTTLE        4      3.75
##   Country
## 1 Austria
## 2 Austria
## 3 Austria
## 4 Austria
## 5 Austria
## 6 Austria
```

V zgornji tabeli lahko opazimo, da je v podatkih kvantiteta v prvi vrstici negativna, kar je mogoče tudi napaka v podatkih. R takšnih napak seveda avtomatsko ne more odkriti, si pa lahko napišemo ukaze, ki bodo zaznali te nepravilnosti.

Najprej si poglejmo kako bi ugotovili katere vrednosti so negativne.

```
logicni_vektor <- podatki$Quantity < 0
head(logicni_vektor)
```

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

Zgornja ukaza preverita vse vrednosti *Quantity* in za vsako vrednost, ki je negativna, vrneta **TRUE**. Nas pa zanima, če je vsaj ena vrednost **TRUE**. To lahko preprosto naredimo z vgrajeno funkcijo `any()`, ki vrne **TRUE**, če je katerakoli vrednost v podanem vektorju **TRUE**. Sorodno deluje `all()`, ki vrne **TRUE**, če so vse podane vrednosti **TRUE**.

Poglejmo si to na primerih.

```
any(FALSE, FALSE, FALSE)
```

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

```
all(FALSE, FALSE, FALSE)
```

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

```
any(TRUE, TRUE, FALSE)
```

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

```r
all(TRUE, TRUE, FALSE)
```

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

```r
any(TRUE, TRUE, TRUE)
```

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

```r
all(TRUE, TRUE, TRUE)
```

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

```r
any(podatki$Quantity < 0)
```

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

To sta uporabni funkciji za varno delo z **if** stavkom, ker drugače nam bo R zajamral, da zna uporabiti le prvo vrednost vektorja.

```r
if (podatki$Quantity < 0) {
  print("Opozorilo: Podatki vsebujejo negativne količine.")
}
```

```
## Warning in if (podatki$Quantity < 0) {: the condition has length > 1 and only
## the first element will be used
```

```
## [1] "Opozorilo: Podatki vsebujejo negativne količine."
```

S funkcijo `any()` pa deluje brez opozorila.

```r
if (any(podatki$Quantity < 0)) {
  print("Opozorilo: Podatki vsebujejo negativne količine.")
}
```

```
## [1] "Opozorilo: Podatki vsebujejo negativne količine."
```

Preverimo še, če so mogoče negativne tudi cene izdelkov.

```r
if (any(podatki$UnitPrice < 0)) {
  print("Opozorilo: Podatki vsebujejo negativne cene.")
}
```

Na izpisu ni ničesar, ker se je ukaz `print()` preskočil. Če želimo obvestilo, da je bilo vse OK lahko uporabimo **if-else** stavek.

```r
if (<pogoj>) {
  # Ukazi, ki se poženejo, če velja pogoj.
} else {
  # Ukazi, ki se poženejo, če ne velja pogoj.
}
```

V našem primeru bi to uporabili na naslednji način.

```
if (any(podatki$UnitPrice < 0)) {
  print("Opozorilo: Podatki vsebujejo negativne cene.")
} else {
  print("Cene so OK.")
}
```

```
## [1] "Cene so OK."
```

Ukaz for (zanka)

Kot motivacijski primer smo podali branje iz Excelove datoteke z večimi listi. Poglejmo najprej, kako bi zadevo rešili z znanjem, ki smo ga pridobili do sedaj. Preberimo vse podatke v datoteki *online-retail.xlsx* in jih shranimo v enoten `data.frame`. Najprej preberimo podatke.

```
podatki_aus <- read.xlsx("./data_raw/online-retail.xlsx", 1)
podatki_ita <- read.xlsx("./data_raw/online-retail.xlsx", 2)
podatki_gre <- read.xlsx("./data_raw/online-retail.xlsx", 3)
head(podatki_aus)
```

```
##   InvoiceNo StockCode                         Description Quantity UnitPrice
## 1   C538971     22153     ANGEL DECORATION STARS ON DRESS      -48      0.42
## 2    539330     37449  CERAMIC CAKE STAND + HANGING CAKES        8      8.50
## 3    539330     37446  MINI CAKE STAND WITH HANGING CAKES        8      1.45
## 4    539330     22962               JAM JAR WITH PINK LID       12      0.85
## 5    539330     21428 SET3 BOOK BOX GREEN GINGHAM FLOWER         4      4.25
## 6    539330     22113         GREY HEART HOT WATER BOTTLE        4      3.75
##   Country
## 1 Austria
## 2 Austria
## 3 Austria
## 4 Austria
## 5 Austria
## 6 Austria
```

```
head(podatki_ita)
```

```
##   InvoiceNo StockCode                         Description Quantity UnitPrice
## 1    537022     22791        T-LIGHT GLASS FLUTED ANTIQUE       12      1.25
## 2    537022     21287       SCENTED VELVET LOUNGE CANDLE        12      1.25
## 3    537022     79337       BLUE FLOCK GLASS CANDLEHOLDER        6      1.65
## 4    537022     85111 SILVER GLITTER FLOWER VOTIVE HOLDER       12      1.25
## 5    537022     85038     6 CHOCOLATE LOVE HEART T-LIGHTS        6      2.10
## 6    537022     22809             SET OF 6 T-LIGHTS SANTA        6      2.95
##   Country
## 1   Italy
## 2   Italy
## 3   Italy
## 4   Italy
## 5   Italy
## 6   Italy
```

```
head(podatki_gre)
```

```
##   InvoiceNo StockCode                          Description Quantity UnitPrice
## 1    541932     22699    ROSES REGENCY TEACUP AND SAUCER        24      2.55
## 2    541932     22697     GREEN REGENCY TEACUP AND SAUCER       24      2.55
## 3    541932     22957 SET 3 PAPER VINTAGE CHICK PAPER EGG       24      2.95
## 4    541932     22720   SET OF 3 CAKE TINS PANTRY DESIGN        24      4.25
## 5    541932    72760B   VINTAGE CREAM 3 BASKET CAKE STAND       16      8.49
## 6    541932     22763             KEY CABINET MA CAMPAGNE       12      8.50
##   Country
## 1  Greece
## 2  Greece
## 3  Greece
## 4  Greece
## 5  Greece
## 6  Greece
```

Sedaj imamo 3 `data.frame` z enakimi stolpci. Lahko jih združimo v enoten `data.frame` z ukazom `rbind`.

```
podatki <- rbind(podatki_aus, podatki_ita, podatki_gre)
head(podatki)
```

```
##   InvoiceNo StockCode                          Description Quantity UnitPrice
## 1   C538971     22153     ANGEL DECORATION STARS ON DRESS      -48      0.42
## 2    539330     37449  CERAMIC CAKE STAND + HANGING CAKES        8      8.50
## 3    539330     37446  MINI CAKE STAND WITH HANGING CAKES        8      1.45
## 4    539330     22962               JAM JAR WITH PINK LID       12      0.85
## 5    539330     21428 SET3 BOOK BOX GREEN GINGHAM FLOWER         4      4.25
## 6    539330     22113         GREY HEART HOT WATER BOTTLE        4      3.75
##   Country
## 1 Austria
## 2 Austria
## 3 Austria
## 4 Austria
## 5 Austria
## 6 Austria
```

Enako bi dosegli, v kolikor bi zaporedoma dodajali vsak `data.frame`, na primer:

```
podatki <- rbind(podatki_aus, podatki_ita)
podatki <- rbind(podatki, podatki_gre)
```

Tak način nam bo prišel prav kasneje.

Da smo prebrali vse podatke smo morali ročno zapisati po eno vrstico za vsak list v Excelovi datoteki in primerno poimenovati vsaki spremenljivko. Nato smo morali vse te `data.frame` še zapisati v ukaz `rbind`. Sicer je zaenkrat bila to popolnoma sprejemljiva rešitev. Ampak ali bomo res vse to ponavljali za večjo datoteko, ki ima 37 listov? Kaj pa, če bomo imeli datoteko s 100 listi ali več? Obstajati mora boljša rešitev!

R nam omogoča, da se ponovnemu pisanju lahko izognemo z uporabo **for** zanke. For zanka v R-ju izgleda takole.

```
for (i in <vektor>) {
  # Ukazi, ki se ponavljajo.
}
```

Zgornji primer ponovi vse ukaze znotraj zavitih oklepajev {} za vsako vrednost v **<vektor>**. Eni ponovitvi z drugimi besedami pravimo **iteracija** zanke. Poglejmo si preprost primer.

```
for (i in 1:3) {
  print("Ponovitev!")
}
```

```
## [1] "Ponovitev!"
## [1] "Ponovitev!"
## [1] "Ponovitev!"
```

Vsaka ponovitev zanke ni popolnoma enaka. V vsaki ponovitvi je drugačna vrednost spremenljivke *i*. To lahko ponazorimo z naslednjim primerom.

```
for (i in 1:3) {
  print("Ponovitev!")
  print(i)
}
```

```
## [1] "Ponovitev!"
## [1] 1
## [1] "Ponovitev!"
## [1] 2
## [1] "Ponovitev!"
## [1] 3
```

Za primer lahko z zanko izračunamo vsoto neke množice števil:

```
x          <- c(2, 4, 7, 5)
moja_vsota <- 0
for (i in x) {
  print("i: ")
  print(i)
  moja_vsota <- moja_vsota + i
  print("moja_vsota: ")
  print(moja_vsota)
  print("-----")
}
```

```
## [1] "i: "
## [1] 2
## [1] "moja_vsota: "
## [1] 2
## [1] "-----"
## [1] "i: "
## [1] 4
## [1] "moja_vsota: "
```

```
## [1] 6
## [1] "-----"
## [1] "i: "
## [1] 7
## [1] "moja_vsota: "
## [1] 13
## [1] "-----"
## [1] "i: "
## [1] 5
## [1] "moja_vsota: "
## [1] 18
## [1] "-----"
```

```
moja_vsota
```

```
## [1] 18
```

```
sum(x)
```

```
## [1] 18
```

Sicer v R poznamo ukaz `sum`, ki to naredi veliko preprosteje. Ampak v bistvu računalnik v ozadju tudi pri tej funkciji uporabi for zanko, le da je mi ne vidimo. Tako se večina računanja v računalnikih v bistvu izvaja z zankami. Sedaj imamo potrebno znanje, ki ga potrebujemo, da preberemo več listov v Excelu z zanko. Preberimo torej sedaj podatke iz večje datoteke *online-retail-large.xlsx*. Imamo 37 listov. Poizkusimo z

```
for (i in 1:37) {
  podatki <- read.xlsx("./data_raw/online-retail-large.xlsx", sheet = i)
}
```

Opazimo težavo. V tem primeru se bodo podatki vedno shranili v isto spremenljivko in jih na koncu ne bomo morali enostavno združiti. Na srečo lahko uporabimo `rbind` kar znotraj zanke! Najprej moramo ustvariti prazen `data.frame`, kateremu nato dodajamo vrstice.

```
podatki <- NULL  # To nam samo rezervira mesto za to spremenljivko.
for (i in 1:37) {
  podatki_tmp <- read.xlsx("./data_raw/online-retail-large.xlsx", sheet = i)
  podatki     <- rbind(podatki, podatki_tmp)
}
```

S funkcijo `View` lahko preverimo, ali podatki izgledajo v redu.

Naš pristop pa ima še eno pomanjkljivost in sicer to, da moramo vnaprej prešteti število listov v Excelu. S tem se lahko spoprimemo s funkcijo `getSheetNames`, ki nam bo vrnila vektor imen listov v Excelu.

```
imena_listov <- getSheetNames("./data_raw/online-retail-large.xlsx")
imena_listov
```

```
##  [1] "France"               "Australia"            "Netherlands"
##  [4] "Germany"              "Norway"               "EIRE"
##  [7] "Switzerland"          "Spain"                "Poland"
## [10] "Portugal"             "Italy"                "Belgium"
```

```
## [13] "Lithuania"            "Japan"                "Iceland"
## [16] "Channel Islands"      "Denmark"              "Cyprus"
## [19] "Sweden"               "Austria"              "Israel"
## [22] "Finland"              "Bahrain"              "Greece"
## [25] "Hong Kong"            "Singapore"            "Lebanon"
## [28] "United Arab Emirates" "Saudi Arabia"         "Czech Republic"
## [31] "Canada"               "Unspecified"          "Brazil"
## [34] "USA"                  "European Community"   "Malta"
## [37] "RSA"
```

Namesto uporabe številskih indeksov v for zanki, lahko uporabimo tudi vektor besed. V našem primeru bodo to imena listov.

```
podatki <- NULL
for (ime_lista in imena_listov) {
  podatki_tmp <- read.xlsx("./data_raw/online-retail-large.xlsx",
                           sheet = ime_lista)
  podatki     <- rbind(podatki, podatki_tmp)
}
```

Včasih želimo kakšno ponovitev zanke izpustiti. Na primer, opazimo, da imamo med listi v Excelu navedeno državo "Unspecified". Recimo, da tega podatka ne želimo v našem `data.frame`. Seveda bi lahko enostavno prebrali vse podatke in nato ustrezne vrstice izbrisali. Lahko pa branje tega lista enostavno izpustimo iz izvajanja zanke. Za to lahko uporabimo kombinacijo pogojnega stavka, ki bo preveril, ali je ime lista enako "Unspecified" in posebnega ukaza `next`, ki preskoči izvedbo ponovitve zanke.

```
podatki <- NULL
for (ime_lista in imena_listov) {
  if (ime_lista == "Unspecified") {
    print("Ime lista je Unspecified. To izvedbo zanke bom preskočil.")
    next
  }
  podatki_tmp <- read.xlsx("./data_raw/online-retail-large.xlsx",
                           sheet = ime_lista)
  podatki     <- rbind(podatki, podatki_tmp)
}
```

```
## [1] "Ime lista je Unspecified. To izvedbo zanke bom preskočil."
```

Poglejmo sedaj, ali imamo te podatke slučajno v `data.frame`.

```
podatki[podatki$Country == "Unspecified", ]
```

```
## [1] InvoiceNo   StockCode   Description Quantity    UnitPrice   Country
## <0 rows> (or 0-length row.names)
```

Vidimo, da podatkov za to državo nimamo.

Podatke smo primerno naložili v R in sedaj imamo `data.frame` s katerim lahko delamo naprej. Nad tem `data.frame` lahko sedaj izvajamo manipulacije, filtriranje in podobno. Recimo, da nas zanima, koliko prihodka ima podjetje v kateri državi. S tem lahko na primer identificiramo potencialne tržne niše in ustrezno usmerimo marketing in naše vire. Poglejmo si naš `data.frame`.

```r
head(podatki)
```

```
##   InvoiceNo StockCode                      Description Quantity UnitPrice
## 1    536370     22728       ALARM CLOCK BAKELIKE PINK       24      3.75
## 2    536370     22727       ALARM CLOCK BAKELIKE RED        24      3.75
## 3    536370     22726      ALARM CLOCK BAKELIKE GREEN       12      3.75
## 4    536370     21724 PANDA AND BUNNIES STICKER SHEET       12      0.85
## 5    536370     21883                STARS GIFT TAPE        24      0.65
## 6    536370     10002     INFLATABLE POLITICAL GLOBE        48      0.85
##   Country
## 1  France
## 2  France
## 3  France
## 4  France
## 5  France
## 6  France
```

Vsaka vrstica nam prikazuje število določenega artikla in ceno za enoto artikla. Nas zanima skupen prihodek, tako da moramo najprej ustvariti nov stolpec, kjer bomo pomnožili ti dve vrednosti.

```r
podatki$Total_sales <- podatki$Quantity * podatki$UnitPrice
head(podatki)
```

```
##   InvoiceNo StockCode                      Description Quantity UnitPrice
## 1    536370     22728       ALARM CLOCK BAKELIKE PINK       24      3.75
## 2    536370     22727       ALARM CLOCK BAKELIKE RED        24      3.75
## 3    536370     22726      ALARM CLOCK BAKELIKE GREEN       12      3.75
## 4    536370     21724 PANDA AND BUNNIES STICKER SHEET       12      0.85
## 5    536370     21883                STARS GIFT TAPE        24      0.65
## 6    536370     10002     INFLATABLE POLITICAL GLOBE        48      0.85
##   Country Total_sales
## 1  France        90.0
## 2  France        90.0
## 3  France        45.0
## 4  France        10.2
## 5  France        15.6
## 6  France        40.8
```

Preostane nam samo še izračun vsote stolpca `Total_sales` za vsako državo. V R obstaja za to funkcija `aggregate`. Tej funkciji moramo podati 3 parametre:

- `x`. Vektor, katerega želimo agregirati. V našem primeru je to stolpec `Total_sales`.
- `by`. Vektor, glede na katerega želimo agregirati. V našem primeru je to stolpec `Country`. Vrednosti v ta parameter moramo podati v obliki seznama. To naredimo tako, da enostavno uporabimo klic `list(<vektor>)`, kjer je `<vektor>` ime vektorja, po katerem želimo agregirati. Lahko jih uporabimo tudi več naenkrat.
- `FUN`. Katero funkcijo želimo uporabiti. V našem primeru bo to funkcija `sum`.

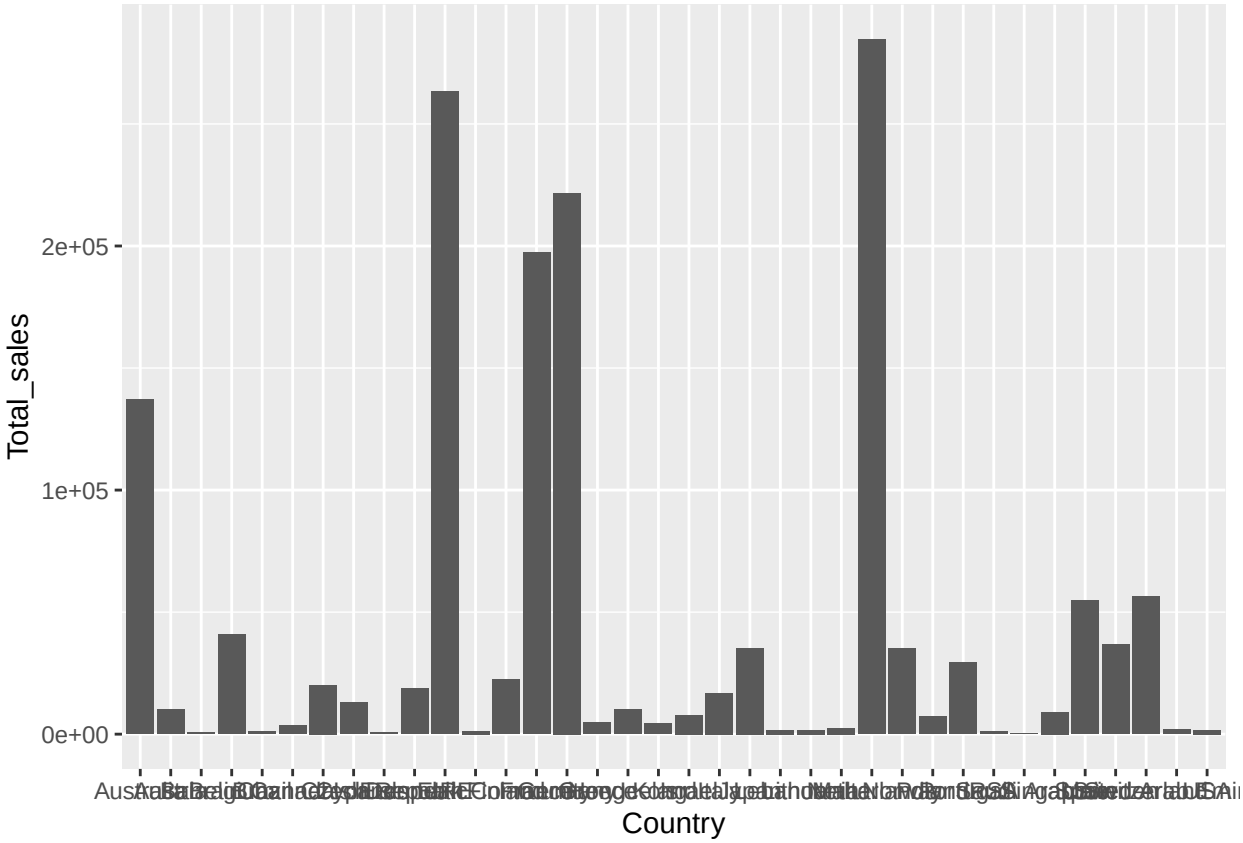
Uporabimo sedaj to funkcijo in rezultate shranimo v novo spremenljivko.

```
podatki_sum <- aggregate(x = podatki$Total_sales, by = list(podatki$Country), FUN = sum)
podatki_sum
```

```
##                 Group.1         x
## 1             Australia 137077.27
## 2               Austria  10154.32
## 3               Bahrain    548.40
## 4               Belgium  40910.96
## 5                Brazil   1143.60
## 6                Canada   3666.38
## 7       Channel Islands  20086.29
## 8                Cyprus  12946.29
## 9         Czech Republic    707.72
## 10               Denmark  18768.14
## 11                  EIRE 263276.82
## 12   European Community   1291.75
## 13               Finland  22326.74
## 14                France 197403.90
## 15               Germany 221698.21
## 16                Greece   4710.52
## 17             Hong Kong  10117.04
## 18               Iceland   4310.00
## 19                Israel   7907.82
## 20                 Italy  16890.51
## 21                 Japan  35340.62
## 22               Lebanon   1693.88
## 23             Lithuania   1661.06
## 24                 Malta   2505.47
## 25           Netherlands 284661.54
## 26                Norway  35163.46
## 27                Poland   7213.14
## 28              Portugal  29367.02
## 29                   RSA   1002.31
## 30          Saudi Arabia    131.17
## 31             Singapore   9120.39
## 32                 Spain  54774.58
## 33                Sweden  36595.91
## 34           Switzerland  56385.35
## 35 United Arab Emirates   1902.28
## 36                   USA   1730.92
```

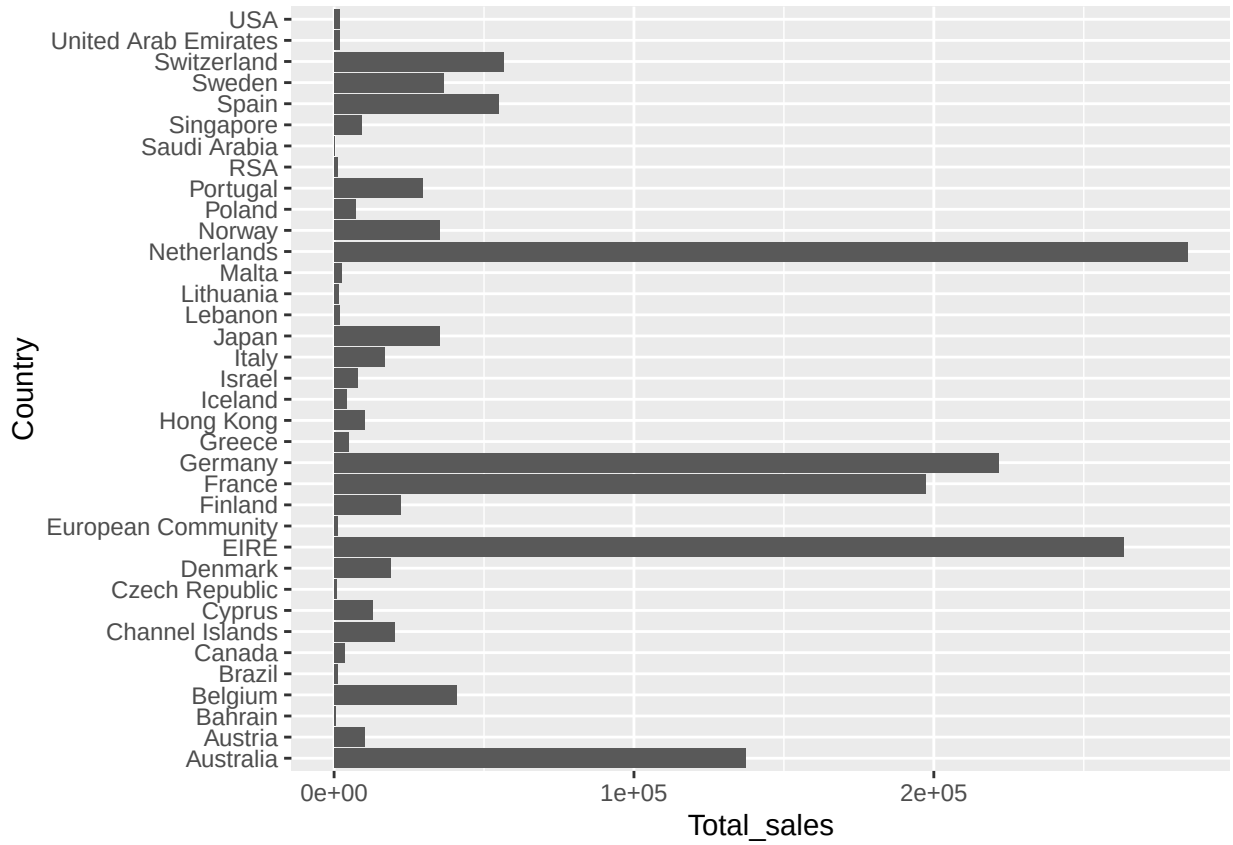
Za lepši prikaz zadevo še vizualizirajmo. Pred tem še spremenimo imena stolpcev.

```
colnames(podatki_sum) <- c("Country", "Total_sales")
library(ggplot2)
ggplot(data = podatki_sum, aes(x = Country, y = Total_sales)) +
  geom_bar(stat = "identity")
```



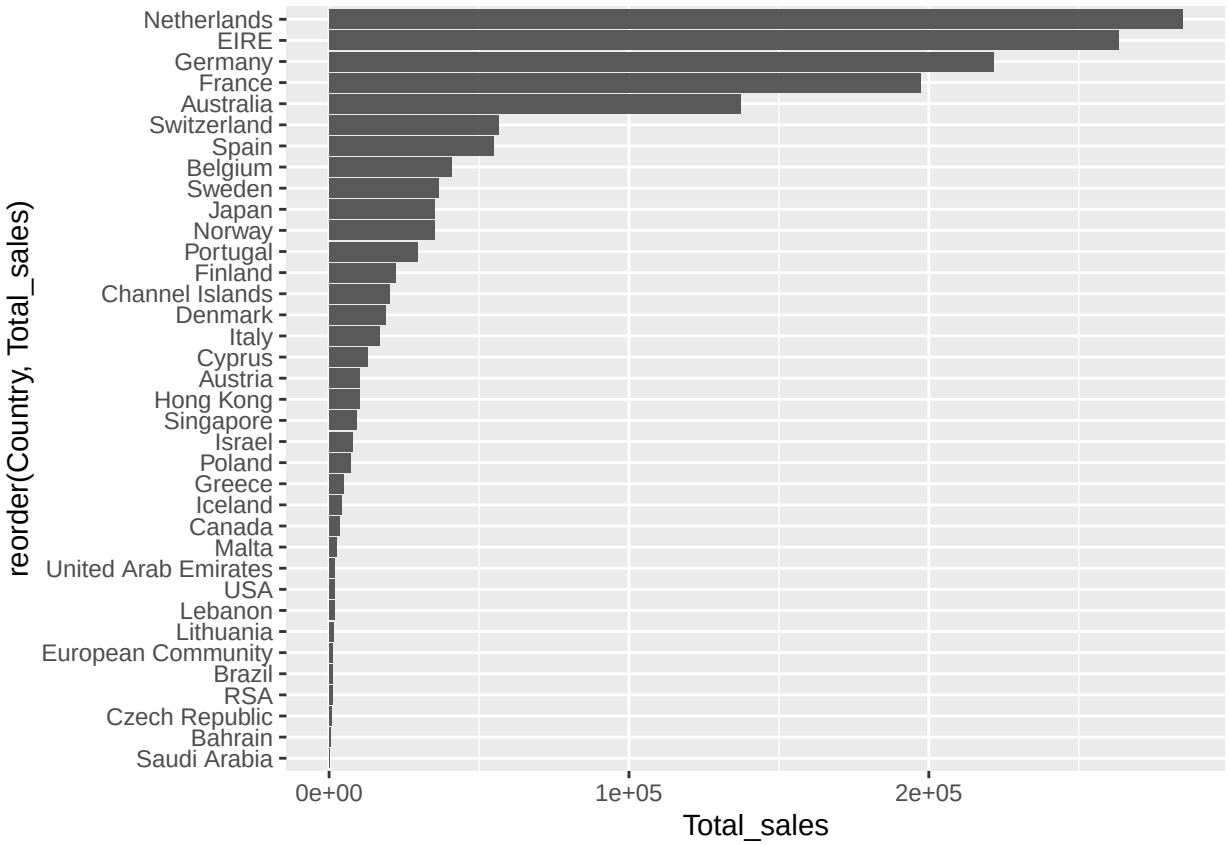
Za lepši prikaz lahko uporabimo še `coord_flip`, ki ga enostavno dodamo našemu grafu. Ta bo zamenjal osi x in y.

```
ggplot(data = podatki_sum, aes(x = Country, y = Total_sales)) +
  geom_bar(stat = "identity") +
  coord_flip()
```



Poizkusimo še urediti stolpce po velikosti. To najlažje naredimo tako, da uporabimo funkcijo `reorder` znotraj estetik.

```
ggplot(data = podatki_sum, aes(x = reorder(Country, Total_sales), y = Total_sales)) +
  geom_bar(stat = "identity") +
  coord_flip()
```



Ukaz break

Namesto, da ponovitev zanke preskočimo, lahko na določeni točki izvajanje zanke tudi prekinemo (na primer če pridemo do kakšnih nesmiselnih vrednosti). V tem primeru namesto `next` uporabimo ukaz `break`. Poglejmo si še to na preprostem primeru. Recimo, da imamo nek vektor pozitivnih in negativnih števil. Najprej izračunajmo vsoto pozitivnih s for zanko.

```
x          <- c(2, 5, -2, -4, 3, -1)
moja_vsota <- 0
for (i in x) {
  print("i: ")
  print(i)
  if (i < 0) {
    print("i je manjši od 0 zato bom izpustil to iteracijo.")
    next
  }
  moja_vsota <- moja_vsota + i
  print("moja_vsota: ")
  print(moja_vsota)
  print("-----")
}
```

```
## [1] "i: "
## [1] 2
```

```
## [1] "moja_vsota: "
## [1] 2
## [1] "-----"
## [1] "i: "
## [1] 5
## [1] "moja_vsota: "
## [1] 7
## [1] "-----"
## [1] "i: "
## [1] -2
## [1] "i je manjši od 0 zato bom izpustil to iteracijo."
## [1] "i: "
## [1] -4
## [1] "i je manjši od 0 zato bom izpustil to iteracijo."
## [1] "i: "
## [1] 3
## [1] "moja_vsota: "
## [1] 10
## [1] "-----"
## [1] "i: "
## [1] -1
## [1] "i je manjši od 0 zato bom izpustil to iteracijo."
```

```
moja_vsota
```

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

Sedaj pa zanko prekinimo, ko pridemo do prvega negativnega števila.

```
x          <- c(2, 5, -2, -4, 3, -1)
moja_vsota <- 0
for (i in x) {
  print("i: ")
  print(i)
  if (i < 0) {
    print("i je manjši od 0 zato bom prekinil izvajanje zanke.")
    break
  }
  moja_vsota <- moja_vsota + i
  print("moja_vsota: ")
  print(moja_vsota)
  print("-----")
}
```

```
## [1] "i: "
## [1] 2
## [1] "moja_vsota: "
## [1] 2
## [1] "-----"
## [1] "i: "
## [1] 5
## [1] "moja_vsota: "
## [1] 7
```

```
## [1] "-----"
## [1] "i: "
## [1] -2
## [1] "i je manjši od 0 zato bom prekinil izvajanje zanke."
```

```
moja_vsota
```

```
## [1] 7
```

Seznami

Oglejmo si še podatkovno strukturo, ki je primerna za hranjenje različnih tipov spremenljivk. Struktura je podobna **vektorju** in se imenuje **seznam** oziroma **list**. **List** se od **vektorja** razlikuje predvsem po tem, da lahko hrani različne tipe spremenljivk in tudi druge strukture različnih dolžin.

```
seznam <- list(1, "dva", c(TRUE, FALSE), c(5,6,7), "šest")
seznam
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] "dva"
##
## [[3]]
## [1]  TRUE FALSE
##
## [[4]]
## [1] 5 6 7
##
## [[5]]
## [1] "šest"
```

Indeksiranje seznama je podobno kot pri vektorjih le da uporabljamo dvojne oglate oklepaje `[[]]`.

```
seznam[[2]]
```

```
## [1] "dva"
```

Vrednosti seznama lahko tudi poimenujemo podobno, kot stolpce `data.frame`.

```
seznam <- list(stevilo1 = 1, niz1 = "dva", bool1 = c(TRUE, bool = FALSE),
               vektor = c(5,6,7), niz2 = "šest")
seznam[['niz2']]
```

```
## [1] "šest"
```

```
seznam[['bool1']]
```

```
##        bool
##  TRUE FALSE
```

Seznamu lahko tudi preprosto dodajamo nove vredosti.

```
seznam[['stevilo2']] <- 5
seznam[[7]] <- 'konec'
seznam
```

```
## $stevilo1
## [1] 1
##
## $niz1
## [1] "dva"
##
## $bool1
##        bool
##  TRUE FALSE
##
## $vektor
## [1] 5 6 7
##
## $niz2
## [1] "šest"
##
## $stevilo2
## [1] 5
##
## [[7]]
## [1] "konec"
```

Vsak element seznama lahko shrani tudi celotno tabelo. Na primer več listov Excela lahko shranimo kot elemente seznama:

```
podatki <- list()  # Ustvarimo prazen seznam.
podatki[[1]] <- read.xlsx("./data_raw/online-retail.xlsx", 1)
podatki[[2]] <- read.xlsx("./data_raw/online-retail.xlsx", 2)
podatki[[3]] <- read.xlsx("./data_raw/online-retail.xlsx", 3)
head(podatki[[1]])
```

```
##   InvoiceNo StockCode                         Description Quantity UnitPrice
## 1   C538971     22153     ANGEL DECORATION STARS ON DRESS      -48      0.42
## 2    539330     37449  CERAMIC CAKE STAND + HANGING CAKES        8      8.50
## 3    539330     37446  MINI CAKE STAND WITH HANGING CAKES        8      1.45
## 4    539330     22962               JAM JAR WITH PINK LID       12      0.85
## 5    539330     21428 SET3 BOOK BOX GREEN GINGHAM FLOWER         4      4.25
## 6    539330     22113         GREY HEART HOT WATER BOTTLE        4      3.75
##   Country
## 1 Austria
## 2 Austria
## 3 Austria
## 4 Austria
## 5 Austria
## 6 Austria
```

```
head(podatki[[2]])
```

```
##   InvoiceNo StockCode                          Description Quantity UnitPrice
## 1    537022     22791         T-LIGHT GLASS FLUTED ANTIQUE       12      1.25
## 2    537022     21287        SCENTED VELVET LOUNGE CANDLE        12      1.25
## 3    537022     79337        BLUE FLOCK GLASS CANDLEHOLDER        6      1.65
## 4    537022     85111 SILVER GLITTER FLOWER VOTIVE HOLDER       12      1.25
## 5    537022     85038     6 CHOCOLATE LOVE HEART T-LIGHTS        6      2.10
## 6    537022     22809              SET OF 6 T-LIGHTS SANTA        6      2.95
##   Country
## 1   Italy
## 2   Italy
## 3   Italy
## 4   Italy
## 5   Italy
## 6   Italy
```

```
head(podatki[[3]])
```

```
##   InvoiceNo StockCode                          Description Quantity UnitPrice
## 1    541932     22699     ROSES REGENCY TEACUP AND SAUCER        24      2.55
## 2    541932     22697      GREEN REGENCY TEACUP AND SAUCER       24      2.55
## 3    541932     22957 SET 3 PAPER VINTAGE CHICK PAPER EGG       24      2.95
## 4    541932     22720   SET OF 3 CAKE TINS PANTRY DESIGN        24      4.25
## 5    541932    72760B   VINTAGE CREAM 3 BASKET CAKE STAND       16      8.49
## 6    541932     22763              KEY CABINET MA CAMPAGNE       12      8.50
##   Country
## 1  Greece
## 2  Greece
## 3  Greece
## 4  Greece
## 5  Greece
## 6  Greece
```

Dodatek: analiza stroškov gospodinjstva

Da utrdimo znanje o zankah si poglejmo še en primer. V datoteki *stroskiDelavnica.xlsx* imamo zapisane stroške gospodinjstva. Vsak list predstavlja svoj mesec. Prvi list ima opozorilo, da se poleg treh stalnih stroškov včasih pojavi tudi strošek za plin. Naložimo vse podatke brez prvega lista, ki se imenuje "Opozorilo". Liste bomo za razliko od prejšnjega primera najprej shranili v seznam in jih nato združili, da povadimo še sezname.

```
podatki    <- list()
sheetNames <- getSheetNames("./data_raw/stroskiDelavnica.xlsx")
for (imeLista in sheetNames) {
  if (imeLista == "Opozorilo") {
    next  # Preskočimo prvi list.
  }
  podatki[[imeLista]] <- read.xlsx("./data_raw/stroskiDelavnica.xlsx", imeLista)
}
head(podatki[[1]])
```

```
##          Strosek Vrednost
## 1     Elektrika   121.40
## 2 Komunala+voda    16.00
## 3      Internet    29.03
```

```
length(podatki)
```

```
## [1] 71
```

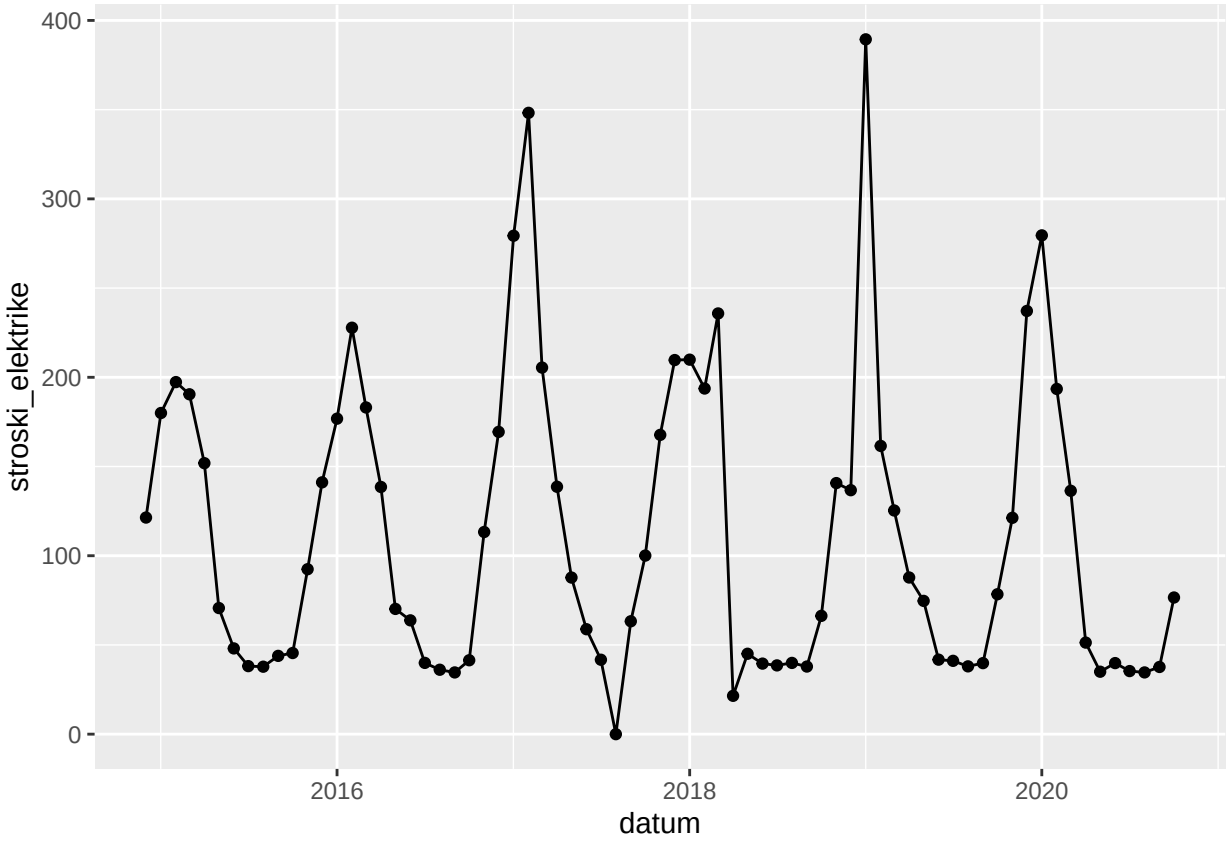
Izrišimo stroške elektrike po mesecih. Najprej naredimo tabelo, ki izloči samo potrebne podatke. Tudi to bomo ustvarili s for zanko.

```
# Ustvarimo prazen data frame.
mesecni_podatki <- data.frame(mesec = character(0),
                              stroski_elektrike = numeric(0))
vrstica <- 1
for (imeLista in sheetNames) {
  if (imeLista == "Opozorilo") {
    next
  }
  podatki_temp <- podatki[[imeLista]]
  # Vstavimo ime lista in podatek o elektriki v data frame (prvi in drugi stolpec).
  mesecni_podatki[vrstica, 1] <- imeLista
  mesecni_podatki[vrstica, 2] <- podatki_temp[podatki_temp$Strosek == "Elektrika",
                                              "Vrednost"]
  # Povečamo spremenljivko vrstica, da bomo v naslednji ponovitvi dodali
  # vrednosti v naslednjo vrstico.
  vrstica <- vrstica + 1
}
# Dodamo datume.
datumi <- seq.Date(as.Date("2014-12-01"), as.Date("2020-10-01"), by = "months")
mesecni_podatki$datum <- datumi
head(mesecni_podatki)
```

```
##           mesec stroski_elektrike      datum
## 1 DECEMBER 2014            121.40 2014-12-01
## 2   JANUAR 2015            179.99 2015-01-01
## 3  FEBRUAR 2015            197.32 2015-02-01
## 4    MAREC 2015            190.48 2015-03-01
## 5    APRIL 2015            151.87 2015-04-01
## 6      MAJ 2015             70.69 2015-05-01
```

Sedaj pa preprosto z `ggplot()` izrišemo krivuljo.

```
library(ggplot2)
ggplot(mesecni_podatki, aes(x = datum, y = stroski_elektrike, group = 1)) +
  geom_point() + geom_line()
```



Domača naloga

a) Preberite podatke v mapi *data_raw* o ameriških volitvah. Podatke smo pobrali 6. novembra 2020 iz: https://www.kaggle.com/unanimad/us-election-2020?select=president_county_candidate.csv. Za vsako zvezno državo posebej izračunajte število glasov, ki sta jih dobila glavna kandidata Joe Biden in Donald Trump in jih izpišite na konzolo. Namig 1: da dobite unikatne vrednosti zveznih držav uporabite funkcijo `unique()`.

Primer rešitve:

```
## [1] "Zvezna država:"
## [1] "Delaware"
## [1] "Joe Biden:"
## [1] 295413
## [1] "Donald Trump:"
## [1] 199857
## [1] "------------------"
## [1] "Zvezna država:"
## [1] "District of Columbia"
## [1] "Joe Biden:"
## [1] 258561
## [1] "Donald Trump:"
## [1] 14449
## [1] "------------------"
```

b) Za vsako zvezno državo posebej izračunajte število glasov, ki so jih dobili kandidati in jih shranite v datoteko *rezultati.csv*. Rezultat zapišite le, če ima kandidat vsaj en glas v zvezni državi. Namig 1: da dobite unikatne vrednosti zveznih držav in kandidatov uporabite funkcijo `unique()`. Namig 2: najlažje bo to rešiti z dvema for zankama, kjer naj bo ena znotraj druge. **Težja!**

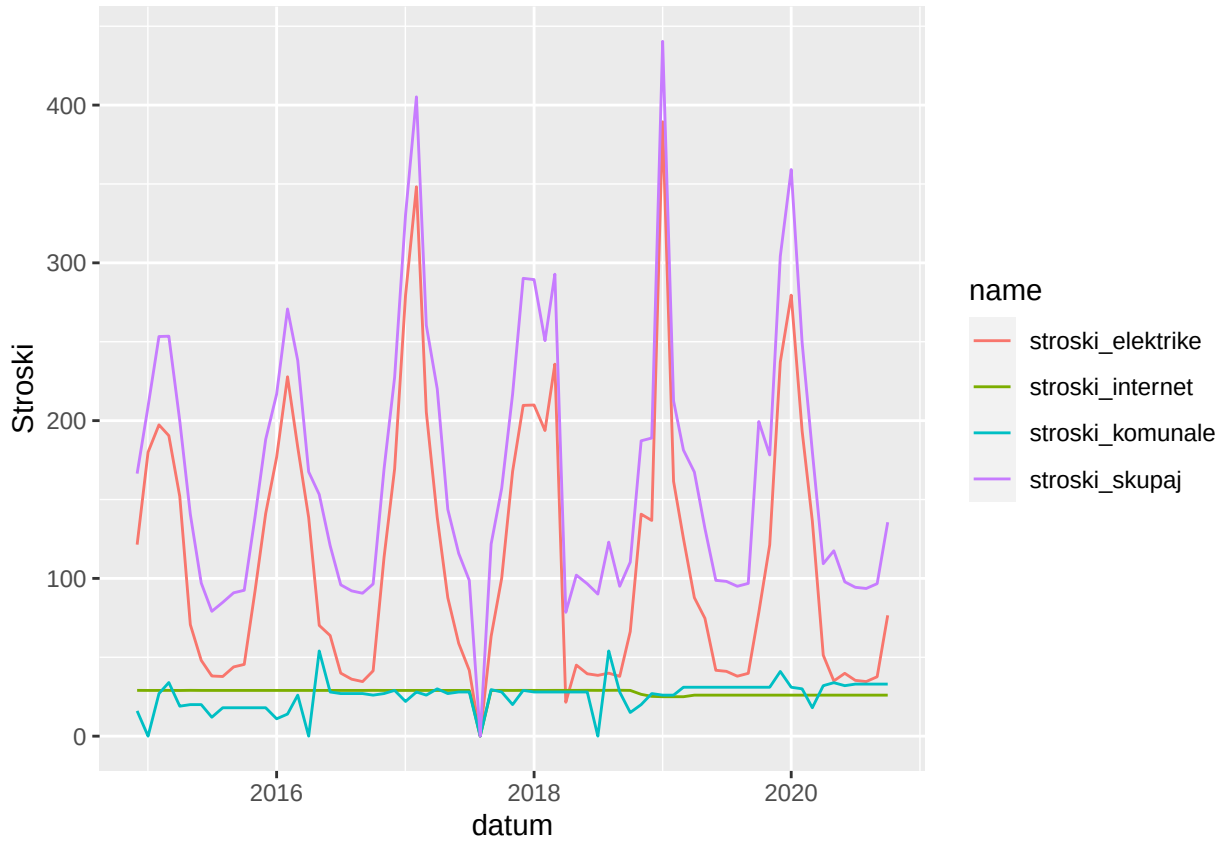
Primer shranjene tabele:

```
##                 drzava      kandidat glasovi
## 1             Delaware     Joe Biden  295413
## 2             Delaware  Donald Trump  199857
## 3             Delaware  Jo Jorgensen    4979
## 4             Delaware Howie Hawkins    2135
## 5 District of Columbia     Joe Biden  258561
## 6 District of Columbia  Donald Trump   14449
```

c) Preberite podatke datoteke *stroskiDelavnica.xlsx* in ustvarite tabelo, ki vsebuje poleg stroškov elektrike še stroške za Komunala + voda in Internet ter skupne mesečne stroške.

```
##           mesec stroski_elektrike stroski_komunale stroski_internet
## 1 DECEMBER 2014            121.40               16            29.03
## 2   JANUAR 2015            179.99                0            28.99
## 3  FEBRUAR 2015            197.32               27            28.99
## 4    MAREC 2015            190.48               34            29.02
## 5    APRIL 2015            151.87               19            28.96
## 6      MAJ 2015             70.69               20            29.04
##   stroski_skupaj      datum
## 1         166.43 2014-12-01
## 2         208.98 2015-01-01
## 3         253.31 2015-02-01
## 4         253.50 2015-03-01
## 5         199.83 2015-04-01
## 6         140.63 2015-05-01
```

d) Poskušajte vse štiri stroške tudi izrisati na en graf. Če ne gre, pa naredite štiri posamezne grafe.



Reference

Chen, Daqing, Sai Laing Sain, and Kun Guo. 2012. "Data mining for the online retail industry: A case study of RFM model-based customer segmentation using data mining." *Journal of Database Marketing & Customer Strategy Management* 19 (3): 197–208.