Data-analyysi R-ohjelmistolla, kevät 2015

Viikko 3

Ensimmäisen ja toisen viikon harjoituksissa keskityttiin matriisien ja vektorien käsittelyyn sekä aineiston pintapuoliseen tarkasteluun. Nyt kolmannella viikolla syvennetään otetta aineistoon ja tutkitaan ohjelmoinnin näkökulmasta hieman mutkikkaampia rakenteita.

Tähän asti kaikkea on tehty vain kerran ja nyt tutustutaankin siihen, kuinka jotakin operaatioita saadaan toistettua ja koodia uudelleenkäytettyä. Samalla tutustutaan omien funktioiden kirjoittamiseen ja for-silmukkaan.

Aineiston analysoinnissa jatketaan tunnuslukujen tarkastelua ja opetellaan uusia työkaluja, joiden avulla perusanalyysi saadaan joustavammaksi ja nopeammaksi.

1 Omat funktiot

R:ssä on valmiina funktiot lähes kaikkeen peruskäyttöön, esimerkiksi otoskeskiarvon, -varianssin ja vastaavien laskemiseen. Joskus näitä funktiota ja niiden tuloksia on kuitenkin tarve yhdistellä uusiksi funktioiksi. Hyvin kirjoitettu funktio voidaan myös helposti siirtää uusiin koodeihin ja näin päästään käyttämään jo luotua koodia nopeasti uudestaan.

Katsotaan seuraavaksi miten voidaan tehdä omia funktioita ja miten niitä käytetään.

Esimerkki 1. Luodaan funktio `zeros()`, joka palauttaa `N` kappaletta nollia.

```
zeros <- function(N) {
  z <- rep(0,N)
  return(z)
}
```

Kerrataan vielä pala kerrallaan, mitä edellisessä koodissa tapahtui.

- Ensin määritellään funktion nimi `zeros`.
- Tähän sijoitetaan sijoitusnuolella funktio, jolla on yksi argumentti, `N`.
- Funktion toiminnot kirjoitetaan aaltosulkeiden `{}` sisään
  - Muuttujaan `z` sijoitetaan `N` kappaletta nollia
  - muuttuja `z` palautetaan komennolla `return(z)`

Huomaa: Funktion määrittelevä koodi täytyy ajaa, jotta funktio tallentuu R:n työtilaan. Vasta tämän jälkeen funktiota voi kutsua.

Esimerkki 2. Käytetään edellisen esimerkin funktiota `zeros` luomaan nollavektoreita.

```
> zeros(1)
[1] 0
> zeros(5)
```

```
[1] 0 0 0 0 0
> zeros(20)
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Tehdään seuraavaksi hieman monimutkaisempi funktio

Esimerkki 3. Käytetään toisen viikon tehtävissä esiteltyä funktiota `runif()` laskemaan `N` satunnaislukua väliltä $[a, b]$. Pyöristetään saadut arvot käyttäen funktiota `round()`, tallennetaan tulos vektoriin `int_values` ja palautetaan se:

```
random_integers <- function(N, a, b) {
  values <- runif(n=N, min=a, max=b)
  int_values <- round(values)

  return(int_values)
}
```

Kutsumalla tätä funktiota saadaan satunnaisia kokonaislukuja:

```
> random_integers(10,0,5)
 [1] 5 5 3 2 2 4 4 1 3 2
> random_integers(4,10,12)
[1] 11 12 11 12
```

Huomaa: Funktion argumentit on annettava siinä järjestyksessä, missä ne on funktion määrittelyssä annettu. Tässä tapauksessa järjestys on `N, a, b`.

Huomaa: Poikkeus edelliseen huomioon: Mikäli argumentit nimetään, voidaan niitä kutsua eri järjestyksessäkin:

Esimerkiksi `random_integers(a=0, N=2, b=2)`

2 Koodin osien toistaminen

Joitakin koodin osia on tarve toistaa useita kertoja, eikä saman koodin kirjoittaminen useaan kertaan peräkkäin ole järkevää. Tällöin tarvitaan for -silmukkaa tai jotakin vastaavaa rakennetta. Tutustutaan seuraavaksi kahteen hieman erilaiseen vaihtoehtoon.

Esimerkki 4. Tehdään ensin yksinkertainen tulostussilmukka ja tutkitaan sen toimintaa:

```
for(i in 1:10) {
print(i)
}
```

Mitä for-silmukka tekee? Se toistaa aaltosulkeiden sisälle kirjoitettua koodia. Katsotaan hieman tarkemmin rakennetta:

- `for(i in 1:10)` määrittelee for-silmukan ja sen sisälle muuttujan `i`.
- Aaltosulkeiden sisällä oleva koodi toistetaan siis jokaisella vektorin `1:10` luvulla. Jokaisella näistä toistoista muuttuja `i` saa järjestyksessä yhden tämän vektorin arvoista.

- Silmukan sisällä ei voi tulostaa antamalla pelkän muuttujan nimen komentona, vaan joudutaan käyttämään komentoa `print()`. Harvoin silmukan sisällä kuitenkaan oikeasti halutaan tulostaa.

Tutkitaan nyt mitä käy, kun edellä annetun koodin ajaa:

```
> for(i in 1:10) {
+ print(i)
+ }
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
```

Samat temput voidaan usein tehdä myös komennolla `sapply`, joka palauttaa tuloksen vektorina. Tehdään nyt edellisen esimerkin koodi käyttäen `sapply()`-funktiota.

Esimerkki 5. `sapply()` vaatii funktion kirjoittamisen, mutta funktiot voivat olla usein hyvinkin yksinkertaisia. Annettua funktiota toistetaan jokaisella annetun vektorin arvolla, aivan kuten for-silmukassa. Lopuksi palautetuista arvoista muodostetaan vektori:

```
> sapply(1:10, function(x) { return(x) })
 [1]  1  2  3  4  5  6  7  8  9 10
```

Esimerkki 6. Luodaan matriisi, kirjoitetaan funktio laskemaan sen sarakkeista maksimiarvoja ja lopuksi toistetaan koodi kaikille sarakkeille käyttäen sekä for-silmukkaa että `sapply()`-funktiota.

Luodaan 10x10 matriisi `A`:

```
A <- matrix(1:100, nrow=10)
```

Nyt tehdään funktio, joka valitsee matriisin `A` sarakkeen numero `i` ja palauttaa sen maksimiarvon:

```
max_value <- function(i) {
  return(max(A[,i]))
}
```

Nyt tehdään vektori kaikista maksimiarvoista ensin for-silmukkaa käyttäen:

```
# Tehdään ensin nollavektori tuloksia varten
values <- rep(0,10)

for(j in 1:10){
  # Sijoitetaan saatu arvo values-vektoriin
  values[j] <- max_value(j)
}
```

Nyt `values`-vektori voidaan tulostaa:

```
> values
 [1]  10  20  30  40  50  60  70  80  90 100
```

Tehdään sama vielä käyttäen `sapply()`-funktiota:

```
> sapply(1:10, function(x) max_value(x))
 [1]  10  20  30  40  50  60  70  80  90 100
```

Huomio: Vertaa tätä tulosta viikon 2 harjoitukseen 4d.

Esimerkki 7. Tutustutaan vielä funktioon `apply()`. Tämän funktion avulla voidaan tehdä matriisille operaatioita joko riveittäin tai sarakkeittain.

Luodaan nyt ensin matriisi A, otetaan siitä sarakkeet 4,5,6,7,8 ja tehdään näille laskuja. Funktion `apply()` toisena annettava argumentti (eli `MARGIN=`) määrittelee käytetäänkö rivejä vai sarakkeita: 1 = rivi, 2 = sarake.

```
> A <- matrix(
+   seq(5,9, length.out=100),
+   nrow=10)
> A_1 <- A[,4:8]
> # Riveittäiset keskiarvot
> apply(A_1, 1, mean)
 [1] 7.020202 7.060606 7.101010 7.141414 7.181818 7.222222 7.262626
 [8] 7.303030 7.343434 7.383838
>
> # Sarakkeittaiset summat
> apply(A_1, 2, sum)
[1] 63.93939 67.97980 72.02020 76.06061 80.10101
```

3 Faktorit

Edellisellä viikolla esittelimme lyhyesti R:n tietotyypeistä faktorin. Faktoreiden tarkoitus on R:ssä kuvata luokittelu- tai järjestysasteikollisia muuttujia. Luokitteluasteikollisella muuttujalla tarkoitetaan muuttujaa, jonka arvoilla ei ole selkeää järjestystä. Tällaisia ovat olla esimerkiksi sukupuoli tai kotikunta. Järjestysasteikollisella muuttujalla tarkoitetaan muuttujaa, jonka arvoilla on järjestys, mutta joiden välimatkaa ei voida yksiselitteisesti mitata. Tällaisia muuttujia ovat esimerkiksi pyöräilybarometrin muuttujat Aq6-Aq15, jotka mittaavat vastaajien tyytyväisyyttä erilaisiin Helsingin pyöräilyolosuhteisiin asteikolla Tyytyväinen (1), Melko tyytyväinen (2), Melko tyytymätön (3), Tyytymätön (4). Arvoilla on selkeä järjestys, mutta niillä ei voida laskea, esimerkiksi ei tiedetä onko ero "tyytyväisen"ja "melko tyytyväisen"välillä yhtä suuri kuin ero "melko tyytyväisen"ja "melko tyytymättömän"välillä.

Muuttujan tyypin muuttaminen faktorille kertoo R:lle, että kyseessä on luokittelu- tai järjestysasteikollinen muuttuja, joten se osaa käsitellä sitä oikealla tavalla ainestoja analysoidessa, esimerkiksi regressioanalyysissa. Faktorit ovat hyödyllisiä myös piirrettäessä kuvia ja tehtäessä ristiintaulukoita; tällöin R kirjoittaa luokkien nimet automaattisesti, eikä niitä tarvitse lisätä käsin tai katsoa koodikirjasta.

3.1 Faktorien luominen

Kerrataan faktorin luominen. Luettaessa aineistoa tiedostosta R tekee merkkijonoista automaattisesti faktoreita, jos tiedoston taulukkoon lukevalle funktiolle ei anneta argumentiksi `stringsAsFactors=FALSE`. Aina, kuten esimerkkiaineistomme tapauksessa tämä ei ole toivottua, joten lisäsimme tiedostoa lukiessamme funktioon `read.csv()` argumentin `stringsAsFactors=TRUE`, jolloin merkkijonot (tässä tapauksessa vapaat tekstikentät) säilyivät merkkijonoina. Jos esimerkiksi halutaan luoda faktori merkkijonotyyppisestä muuttujasta, johon on tallennettu opiskelijan pääaine (Matematiikka, Taloustiede tai Tilastotiede), se tapahtuu `factor`-komennolla.

Esimerkki 8.

```
> paa_aine <- c("Mat", "Tal", "Tal", "Til", "Til", "Mat", "Til")
> paa_aine <- factor(paa_aine)
> str(paa_aine)
 Factor w/ 3 levels "Mat","Tal","Til": 1 2 2 3 3 1 3
```

Funktiolla `str()` nähdään, että R muutti muuttujan `paa_aine` koodauksen merkkijonosta:

```
"Mat", "Tal", "Tal", "Til", "Til", "Mat", "Til"
```

numeeriseksi:

```
1 2 2 3 3 1 3
```

Luodulla faktorilla on siis kolme tasoa, `"Mat"`, `"Tal"` ja `"Til"`, eli matematiikan opiskelijat on koodattu uudelleen ykkösiksi, taloustieteen opiskelijat kakkosiksi ja tilastotieteen opiskelijat kolmosiksi.

Aineistossa on usein myös numeerisesti koodattuja luokitteluaineistollisia muuttujia, jotka halutaan muuttaa analyyseja varten faktoreiksi. Otetaan esimerkiksi opiskelijan tiedekuntaa kuvaava muuttuja `tiedekunta`, jonka arvot ovat Matemaattis-luonnontieteellinen tiedekunta (ML), joka on koodattu ykköseksi, ja Valtiotieteellinen tiedekunta (VT), joka on koodattu kakkoseksi. Myös numeerinen muuttuja muutetaan faktoriksi funktiolla `factor()`, mutta nyt funktiolle annetaan argumenttiin `labels` halutut faktorin tasojen kuvaukset merkkijonovektorina.

Esimerkki 9.

```
> tiedekunta <- c(1,2,2,1,1,1,2)
> tiedekunta <- factor(tiedekunta, labels=c("ML", "VT"))
> str(tiedekunta)
 Factor w/ 2 levels "ML","VT": 1 2 2 1 1 1 2
```

Tarkasteltaessa faktoriksi muutettua muuttujaa funktiolla `str()` huomataan, että muuttujan numeeriset arvot pysyivät ennallaan, mutta niihin liitettiin `factor()`-funktiolle annetut tasojen kuvaukset `ML` ja `VT`.

4 Osa-aineistojen valinta, osa 2

4.1 Alkioiden valinta vektorista

Ensimmäisellä viikolla tarkastelimme vektorin alkioiden valintaa. Yksinkertaisimmillaan tämä tapahtuu sijoittamalla valintaehto vektorin indeksiksi. Jos vektorissa on puuttuvia arvoja, lopputulos ei kuitenkaan aina ole toivottu. Yritetään esimerkiksi selvittää kuinka moni vektorin `a` arvoista on alle kuusi `length()`-funktion [1] avulla.

Esimerkki 10.

```
> a <- c(3,NA,6,54,NA,5)
> a < 6
[1]  TRUE    NA FALSE FALSE    NA  TRUE
> a[a<6]
[1]  3 NA NA  5
> length(a[a<6])
[1] 4
```

Huomataan, että vertailu `a<6` palauttaa myös puuttuvan arvon niille `a`:n alkioille, joiden arvo puuttuu. Käytettäessä ehtoa edelleen `a`:n indeksinä, mukaan otetaan niiden `a`:n arvojen lisäksi, jotka ovat pienempiä kuin kuusi, myös puuttuvat arvot. Siten lopputulos on neljän pituinen vektori, joka sisältää kaksi puuttuvaa arvoa halutun kahden pituisen vektorin sijasta. Yksi ratkaisu tähän ongelmaan on soveltaa `which()`-funktiota, joka antaa ne vektorin indeksit, joille argumenttina annettu ehto on tosi, ja jättää automaattisesti puuttuvat arvot huomioimatta.

Esimerkki 11.

```
> which(a<6)
[1] 1 6
> a[which(a<6)]
[1] 3 5
> length(a[which(a<6)])
[1] 2
```

Tässä siis `which()`-funktio palauttaa ensin indeksit, joille `a`:n arvo on pienempää kuin kuusi, ja sijoitettaessa nämä `a`:n indeksiksi saadaan kyseiset arvot, eli 3 ja 5.

Vieläkin suorempi tapa on käyttää `subset()`-funktioita, joka valitsee ensimmäisenä argumenttina annetusta vektorista ne alkiot, joille toisena argumenttina annettu ehto on tosi. Myös `subset()` jättää automaattisesti puuttuvat arvot huomioimatta.

Esimerkki 12.

```
> subset(a, a<6)
[1] 3 5
> length(subset(a, a<6))
[1] 2
```

[1] Komento `sum(a<6, na.rm=T)` olisi tietenkin yksinkertaisempi tapa selvittää asia.

4.2 Osa-aineistojen valinta taulukosta

Tehdään taulukko (kuvitteellisista) R-kurssin osallistujista. Lisätään esimerkkiaineistoomme muuttuja `tutkinto`, joka kuvaa sitä onko opiskelija kandi- vai maisterivaiheessa. Arvotaan vielä opiskelijanumerot ja liitetään muuttujat yhteen `opiskelijat`-taulukoksi.

Esimerkki 13.

```
> tutkinto <- c(1,1,1,NA,2,NA,1)
> tutkinto <- factor(tutkinto, labels=c("kandi", "maisteri"))
> opiskelijaNro <- runif(length(paa_aine), min=1000000, max=1500000)
> opiskelijat <- data.frame(opiskelijaNro, paa_aine, tiedekunta, tutkinto)
> opiskelijat
  opiskelijaNro paa_aine tiedekunta tutkinto
1       1289002      Mat         ML    kandi
2       1404864      Tal         VT    kandi
3       1058537      Tal         VT    kandi
4       1153128      Til         ML     <NA>
5       1315334      Til         ML maisteri
6       1074686      Mat         ML     <NA>
7       1180208      Til         VT    kandi
```

Osasta opiskelijoista ei ole tietoa heidän opiskelujensa vaiheesta, joten `tutkinto`-muuttuja saa puuttuvia arvoja. Oletetaan, että halutaan tarkastella lähemmin kurssin kandivaiheessa olevia opiskelijoita. Naiivi lähestymistapa, jossa valintaehto sijoitetaan `opiskelijat`-taulukon rivi-indeksin paikalle, tuottaa hieman ikävän näköisen lopputuloksen.

Esimerkki 14.

```
> opiskelijat[opiskelijat$tutkinto == 'kandi', ]
     opiskelijaNro paa_aine tiedekunta tutkinto
1          1289002      Mat         ML    kandi
2          1404864      Tal         VT    kandi
3          1058537      Tal         VT    kandi
NA              NA     <NA>       <NA>     <NA>
NA.1            NA     <NA>       <NA>     <NA>
7          1180208      Til         VT    kandi
>
> nrow(opiskelijat[opiskelijat$tutkinto == 'kandi', ])
[1] 6
```

Valittuun osa-aineistoon tulivat mukaan myös rivit, joissa `tutkinto`-muuttujan arvo puuttui, ja kaikki muutkin näiden rivien arvot muuttuivat puuttuviksi [2]. Tämä ei liene haluttu tulos.

Siistimpi lopputulos saadaan joko jälleen `which()`-funktion avulla, tai sitten soveltamalla suoraan `subset()`-funktiota taulukoille. Huomaa, että käytettäessä `subset()`:iä taulukkoon sarakkeen nimeä ei tarvitse kirjoittaa muodossa `opiskelijat$tutkinto`, vaan pelkkä sarakkeen nimi riittää.

[2] Funktio `nrow()` palauttaa argumentiksi annetun matriisin tai taulukon rivien määrän ja `ncol()` taas palauttaa sarakkeiden määrän.

Esimerkki 15.

```
> subset(opiskelijat, tutkinto == 'kandi')
  opiskelijaNro paa_aine tiedekunta tutkinto
1       1289002      Mat         ML    kandi
2       1404864      Tal         VT    kandi
3       1058537      Tal         VT    kandi
7       1180208      Til         VT    kandi
>
> nrow(subset(opiskelijat, tutkinto == 'kandi'))
[1] 4
```

4.3 Tunnuslukujen laskeminen osa-aineistoittain

Liitetään esimerkkiaineistoamme R-kurssin osallistujista vielä muuttuja `opintopisteet`, joka kertoo opiskelijan kurssin alkuun mennessä suorittamat opintopisteet.

Esimerkki 16.

```
> opiskelijat$opintopisteet <- c(20, 33, 55, 120, 230, 172, 50)
> opiskelijat
  opiskelijaNro paa_aine tiedekunta tutkinto opintopisteet
1       1196156      Mat         ML    kandi            20
2       1154825      Tal         VT    kandi            33
3       1072480      Tal         VT    kandi            55
4       1468267      Til         ML     <NA>           120
5       1102544      Til         ML maisteri           230
6       1448363      Mat         ML     <NA>           172
7       1345642      Til         VT    kandi            50
```

Edellisellä viikolla laskimme tunnuslukuja osa-aineistoittain, esimerkiksi keskiät miehille ja naisille erikseen pyöräilybarometri-aineistosta. Tämä onnistuu kätevästi myös R:stä vakiona löytyvällä `apply`-perheen funktiolla `tapply()`. Käytettäessä `tapply()`-funktiota osa-aineistoja ei valita erikseen käsin, vaan `tapply()` tekee sen automaattisesti. Lasketaan esimerkiksi opintopisteiden keskiarvot pääaineittain.

Esimerkki 17.

```
> tapply(opiskelijat$opintopisteet, opiskelijat$paa_aine, mean)
     Mat      Tal      Til
 96.0000  44.0000 133.3333
```

Funktio `tapply()` hajoittaa ensin ensimmäisenä argumenttina annetun vektorin, tässä tapauksessa `opintopisteet`, ryhmiin toisena argumenttina annetun faktorin, tässä tapauksessa `paa_aine`, mukaan. Sen jälkeen `tapply()` suorittaa kolmantena argumenttina annetun funktion, tässä tapauksessa `mean()`:in, kuhunkin osa-aineistoon ja palauttaa tuloksen nimettynä vektorina [3].

[3] Nimetty vektori on muuten kuten tavallinen vektori, mutta sen alkioilla on nimet joilla niihin voidaan viitata.

5 Ristiintaulukointi

5.1 Frekvenssitaulu

Muuttujan jakaumaa voidaan tutkia funktiolla `table()`, joka palauttaa kutakin vektorin alkioden saamaa arvoa kohti, kuinka moni vektorin alkioista saa tämän arvon.

Esimerkki 18.

```
a <- c(10,22,438,7,22,22,10,438,438,22,22,22)
table(a)
a
  7  10  22 438
  1   2   6   3
```

Voidaan esimerkiksi selvittää kuinka moni kurssin osallistujista suorittaa kanditaatin- ja kuinka moni maisterintutkintoa. Tällaista muuttujan eri arvojen määriä kuvaavaa taulua kutsutaan frekvenssitauluksi. Huomaa, että `table()` jättää puuttuvat arvot automaattisesti huomioimatta. Nähdään myös, että R tulostaa taulun sarakkeiden otsikoiksi automaattisesti faktorin tasot, mikä helpottaa tulkintaa.

Esimerkki 19.

```
> table(opiskelijat$tutkinto)

   kandi maisteri
       4        1
```

5.2 Ristiintaulukko

Myös kahden muuttujan yhteisjakauman tarkastelu onnistuu `table()`-funktiolla. Taulua, joka sisältää kahden muuttujan arvojen yhdistelmien aineistossa saamat määrät, kutsutaan *ristiintaulukoksi* (jatkossa myös välillä lyhyemmin *tauluksi*). Taulukoidaan kurssin opiskelijoiden pääaine ja tiedekunta vastakkain.

Esimerkki 20.

```
> table(opiskelijat$paa_aine, opiskelijat$tiedekunta)

      ML VT
  Mat  2  0
  Tal  0  2
  Til  2  1
```

Nähdään että kurssin osallistujista kaikki matemaatikot opiskelevat matemaattis-luonnontieteellisessä tiedekunnassa ja taloustieteilijät valtiotietellisessä. Tilastotieteilijöitä taas opiskelee kummassakin tiedekunnassa.

Ristiintaulukoista pystyy valitsemaan alkoita, rivejä ja sarakkeita kuten matriisista. Valitaan edellisen esimerkin taulun ensimmäisen rivin ensimmäisen sarakkeen alkio, ja sen jälkeen koko ensimmäinen rivi.

Esimerkki 21.

```
> paa_aine_X_tiedekunta <- table(opiskelijat$paa_aine, opiskelijat$tiedekunta)
> paa_aine_X_tiedekunta[1,1]
[1] 2
> paa_aine_X_tiedekunta[1,]
ML VT
 2  0
```

Tauluihin pystyy myös käyttämään funktioita riveittäin ja sarakkeittain `apply()`-funktiolla. Lasketaan esimerkikkitaulun sarakesummat, eli kurssin opiskelijoiden kokonaismäärä tiedekunnittain.

Esimerkki 22.
```
> apply(paa_aine_X_tiedekunta,2,sum)
ML VT
 4  3
```

Rivi- ja sarakeprosentit saadaan funktiolla `prop.table()`, jolle annetaan argumentiksi alkuperäinen taulu, josta prosentit halutaan laskea ja 1 tai 2 sen mukaan, halutaanko rivi- vai sarakeprosentit. Lasketaan esimerkkitaulustamme sarakeprosentit, eli kunkin pääaineen edustajien osuudet tiedekunnittain kurssin osallistujista.

Esimerkki 23.
```
prop.table(paa_aine_X_tiedekunta,2)

            ML        VT
 Mat 0.5000000 0.0000000
 Tal 0.0000000 0.6666667
 Til 0.5000000 0.3333333
```

6 Tehtäviä

6.1 Tehtäviä

1. Harjoitellaan funktioiden tekemistä
   a) Tee funktio `summa()`, joka laskee kaksi lukua yhteen ja palauttaa tuloksen.
   b) Tee funktio `erotus()`, joka vähentää kaksi lukua toisistaan ja palauttaa tuloksen
   c) Laske näiden avulla laskut $4 + 6$ ja $8 - 3$.
2. Harjoitellaan for-silmukan käyttöä summaamiseen.
   a) Tee vektori `values <- c(5,7,9,8,5,4,3,5,7,8,9,2,4)` ja muuttuja `vektorisumma`, jonka arvo on 0.
   b) Määrittele for-silmukka, jossa muuttuja `i` käy läpi arvot `1:length(values)`.
   c) Kirjoita for-silmukan sisään koodi, joka lisää muuttujaan `vektorisumma` vektorin `values` `i`:nnen arvon

d) Laske ja tulosta muuttuja `vektorisumma`. Voit tarkistaa tuloksesi funktiolla `sum()`.

3. Luo matriisi `A <- matrix(1:100, nrow=10)`

   a) Laske jokaisen sarakkeen keskiarvo käyttäen funktiota `sapply()`. Vihje: Esimerkki 6 ja funktio `mean()`.

   b) Laske jokaisen sarakkeen keskiarvo käyttäen funktiota `apply()`. Vihje: Esimerkki 7.

4. Tutkitaan seuraavaksi Polkupyöräbarometria ja opetellaan `apply()`-funktion käyttöä. Näissä voi auttaa katsoa esimerkkiä 7.

   a) Lataa aineisto käyttöön nimelle `pb`, kuten viikolla 2

   b) Valitse taulukosta `pb` sarakkeet `Aq3`, `Aq4` ja `Aq5` (sarakenumerot 11,12,13) ja tallenna ne taulukoksi `pb_tyytyv`.

   c) Muuta taulukosta `pb_tyytyv` 'En osaa sanoa'-vastaukset puuttuviksi

   d) Laske taulukosta `pb_tyytyv` vastaajakohtaiset (riveittäiset) keskiarvot

   e) Laske vielä keskiarvo näistä vastaajakohtaisista keskiarvoista ja pohdi tulosta muutamalla sanalla

   f) *Lisäpohdittavaa innokkaimmille: Miksi lasketaan keskiarvoja ja keskiarvoista keskiarvoja? Tähän ei tarvitse vastata, mutta pieni pohdiskelu voi auttaa.*

5. Tarkastellaan yhden muuttujan jakaumaa ja selvitetään mitä iloa on muuttujien muuttamisesta faktoreiksi.

   a) Tulosta vastaajien pyöräilyaktiivisuutta kuvaavan muuttujan `Aq2` frekvenssitaulu ja visualisoi muuttujaa komennolla `plot(pb$Aq2)` (kuvan ei ole tarkoituskaan olla kauhean järkevä).

   b) Muuta vastaajien pyöräilyaktiivisuutta kuvaaja muuttuja `Aq2` faktoriksi, jonka tasoja ovat sen koodikirjasta löytyvät arvojen kuvaukset. Tason "Päivittäin tai lähes päivittäin"voi lyhentää muotoon "Päivittäin", jotta seuraavassa tehtävässä laskettavat ristiintaulukot mahtuisivat yhdelle riville.

   c) Tulosta faktoriksi muutetun muuttujan `Aq2` frekvenssitaulu ja visualisoi tulos pylväsdiagrammilla (toista a-kohdan `plot`-komento!). Vaihda pylväsdiagrammiin joku mieleisesi väri oletusarvona olevan tylsän harmaan sijasta ja anna diagrammille kuvaava otsikko. Vihje: `col`- ja `main`-argumentit funktiolle `plot`.

6. Puuttuvien arvojen laskeminen ja arvojen muuttaminen puuttuviksi.

   a) Laske kuinka monta puuttuvaa arvoa vastaajien koulutustasoa mittaava muuttuja muuttuja `AG` saa aineistossa?

   b) Entä kuinka monta Muu-arvoa (ks. koodikirja)?

   c) Muuta Muu-arvot puuttuviksi arvoiksi.

d) Tarkista b-kohdan onnistuminen tulostamalla frekvenssitaulu (puuttuvia arvoja ei näy frekvenssitaulussa) ja laskemalla puuttuvat arvot muuttujasta (puuttuvia arvoja pitäisi olla nyt yhtä paljon kuin alunperin + alkuperäiset Muu-arvot).

e) Muuta muuttuja `AG` faktoriksi, jonka tasoja ovat koodikirjasta löytyvät arvojen kuvaukset (tason Muu voi jättää pois, sillä sitä ei enää ole aineistossa).

7. Tutkitaan koulutustason ja pyöräilyaktiivisuuden välistä yhteyttä ristiintaulukoimalla.

   a) Taulukoi kahdessa edellisessä tehtävässä luodut faktoreiksi muutetut muuttujat vastakkain siten, että koulutustaso on taulun riveillä ja pyöräilyaktiivisuus sarakkeilla. Tallenna taulu.

   b) Laske taulusta rivisummat `apply()`- ja `sum()`-funktioiden avulla. Tallenna tulos vektoriin `rivisummat` ja lisää tallenna se alkuperäiseen tauluun `cbind()`-funktiolla.

   c) Laske b-kohdassa muokatusta taulusta sarakesummat `apply()`- ja `sum()`-funktioiden avulla. Tallenna tulos vektoriin `sarakesummat` ja lisää se b-kohdassa muokattuun tauluun `rbind()`-funktiolla.

   d) Tulosta c-kohdan taulu ja tarkasta tulokset käyttämällä `addmargins()`-funktiota alkuperäiseen tauluun. Vihje: Jos haluaa tietää mitä funktio tekee, mitkä ovat sen argumentit, tms., voi ajaa komennon, jossa kirjoitetaan kysymysmerkki funktion nimen eteen, esim. `?addmargins`.

8. Jatkoa edelliseen tehtävään.

   a) Laske edellisen tehtävän alkuperäisestä taulusta riviprosentit, eli muuttujan `Aq2` arvojen osuudet riveittäin käyttämällä `apply()`-funktiota tauluun. Vihje: kirjoita funktio joka jakaa vektorin sen alkioiden summalla ja käytä sitä tauluun `apply()`:llä riveittäin. Näin saat osuudet koulutusryhmittäin.

   b) Tuloksena oleva taulussa rivien ja sarakkeiden paikat vaihtuivat. Käännä se takaisin alkuperäiseen muotoon. Vihje: ensimmäisen viikon matriisioperaatiot. Kerro vielä taulu sadalla, jolloin saat prosentit, ja pyöristä yhden desimaalin tarkkuuteen. Vihje: `round()`-funktio.

   c) Laske edellisen tehtävän alkuperäisestä taulusta sarakeprosentit samalla tavalla kuin riviprosentit ja pyöristä taas yhden desimaalin tarkkuuteen. Tällä kertaa taulu tulee suoraan samoin päin kuin alkuperäinen, eli sitä ei tarvitse kääntää.

   d) Tarkasta edellisten kohtien tulokset käyttämällä `prop.table()`-funktiota alkuperäiseen tauluun.

9. Jatkoa edelliseen tehtävään.

   a) Jos halutaan tarkastella koulutustaustan vaikutusta pyöräilyaktiivisuuteen, kummat edellisessä tehtävän prosenteista ovat mielekkäämmät laskea? Miksi?

b) Tulkitse tulosta. Vaikuttaako koulutustausta mielestäsi pyöräilyaktiivisuuteen (Tähän riittää mielipide, tarkkaan tilastolliseen testaukseen mennä vasta seuraavilla viikoilla kun siihen on käytössä tarvittavat työkalut)?

10. Tunnuslukujen laskemista ryhmittäin

    a) Poista vastaajan työelämästatusta kuvaavasta muuttujasta `AI` arvot "muuten työelämän ulkopuolella"muuttamalla ne puuttuviksi arvoiksi ja tarkasta operaation onnistuminen frekvenssitaulukon avulla samalla tavalla muuttujalle `AG` tehtiin aiemmin.

    b) Muuta muuttuja faktoriksi, tasojen kuvaukset löytyvät koodikirjasta.

    c) Laske keski- ja mediaani-iät erikseen muuttujan eri ryhmille (työtön/opiskelija/eläkeläinen/töissä). Pyöristä keski-iät tasavuosiksi. Vihje: `tapply()`.

11. SU-estimaattorin tarkentuvuus (ks. JTP:n tehtävä 2.4).

    a) Simuloi otokset otoskoilla 5, 10, 25, 50, 100, 250 ja 1000 normaalijakaumasta $N(3, 16)$ ja tallenna otosten keskiarvot vektoriin. Tämän voi tehdä myös `for`-silmukalla, mutta kätevintä lienee käyttää `sapply()`:a. Vihje: aseta halutut otoskoot sisältävä vektori `sapply()`:n ensimmäiseksi argumentiksi.

    Huom. tässä 16 on normaalijakauman varianssi. Tarkasta komennolla `?rnorm` miten normaalijakauma on parametrisoitu R:ssä.

    b) Miten simuloidusta otoksesta laskettujen keskiarvojen ja todellisen jakauman keskiarvoparametrin suhde muuttuu (tai pitäisi keskimäärin muuttua), kun otoskoko kasvaa? Miten selität ilmiötä?