Data-analyysi R-ohjelmistolla, kevät 2015

Viikko 1

Ensimmäisen viikon tavoitteena on saada riittävät taidot käsitellä toisella viikolla käyttöön tulevia aineistoja. Aineistot luetaan R:ään matriiseina tai matriisin tapaisina tietorakenteina, joita kutsutaan data frameiksi. Näiden käsittelyä varten on vektorien ja matriisien käsittelyn oltava hyvin hallussa.

Tehtävien lopuksi tutustumme puuttuvan arvon käsittelyyn. Tämä onkin hieman hankalampaa, sillä puuttuvien arvojen käsittelyä ei koske samat laskutoimitukset, kuin lukuarvoja tai merkkijonoja.

Huomaa! Ensimmäisen viikon tehtävien palautus on vasta toisella viikolla. Tehtävät saa palauttaa aiemminkin, mutta viimeistään 20.3.2015. Lisätietoa tehtävien palautuksesta on kurssin wiki-sivuilla.

1 Skalaarimuuttujat

1.1 Määrittely

Skalaareja voidaan käsitellä R-ohjelmistolla sellaisenaan, syöttämällä haluttu luku konsoliin. Desimaalierottimena toimii tavallinen piste [.].

Esimerkki 1. Tulostetaan ensin lukuja

```
> 15
[1] 15
> 3.14159
[1] 3.14159
```

Muuttuja voidaan sijoittaa lähes minkä tahansa nimiseen muuttujaan käyttämällä sijoitusoperaattoria `<-`. Nimen on syytä kuitenkin alkaa kirjaimella, eikä olemassa olevien funktioiden nimiä ole järkevää tai mahdollista ylikirjoittaa. Tallennetun muuttujan sisällön voi tulostaa antamalla muuttujan nimen komentona.

Esimerkki 2. Sijoitetaan muuttujiin arvoja

```
> a <- 42
> a
[1] 42
> b <- 0.001
> b
[1] 0.001
```

1.2 Laskuoperaatiot

Reaaliluvuilla voidaan suorittaa helposti kaikki peruslaskutoimitukset: yhteen- ja vähennyslasku sekä kerto- ja jakolasku. Laskuoperaatiot voidaan tehdä myös tallennetuille muuttujille, joita käytetään laskun osana aivan kuin numeroita.

Esimerkki 3. Yhteen- ja vähennyslasku

```
> a <- 5
> b <- 3
> a-b
[1] 2
> a+b
[1] 8
> a - 6
[1] -1
```

Esimerkki 4. Kerto- ja jakolasku

```
> a*b
[1] 15
> a/b
[1] 1.666667
> a %% b
[1] 2
```

1.3 Loogiset operaattorit ja totuusarvot

R:ssä on käytössä normaalit vertailuoperaatiot suurempi kuin (`>`), pienempi kuin (`<`), suurempi tai yhtä suuri kuin (`>=`), ja pienempi tai yhtä suuri kuin (`<=`). Yhtäsuuruusvertailu saadaan operaattorilla `==` ja erisuuruusvertailu operaattorilla `!=`. Huom. pelkkä tavallinen yhtäsuuruusmerkki = toimii useimmiten kuten sijoitusoperaattori, joten sitä ei voi käyttää vertailussa.

Vertailuoperaatiot antavat tuloksena totuusarvon `TRUE` tai `FALSE` (nämä voi myös lyhentää kirjaimilla `T` ja `F`). Vertailun tuloksen voi tallentaa muuttujaan.

Esimerkki 5. Vertailuoperaattorien käyttöä:

```
> 1 < 0
[1] FALSE
> x <- 1
> x < 2
[1] TRUE
> 2 == 2
[1] TRUE
> y <- 2 != 2
> y
[1] FALSE
> y == TRUE
[1] FALSE
```

Lisäksi käytössä ovat looginen JA (`&`), TAI (`|`) ja negaatio (`!`), joiden avulla voidaan kirjoittaa pidempiä ehtolauseita.

Esimerkki 6.

```
> (1 > 0) & (1 < 0)
[1] FALSE
> (1 > 0) | (1 < 0)
[1] TRUE
```

```
> !(1 < 0)
[1] TRUE
> TRUE & FALSE
[1] FALSE
> TRUE | FALSE
[1] TRUE
> !FALSE
[1] TRUE
```

2 Vektorit ja matriisit

2.1 Vektorien luominen

Skalaareita, kuten vektoreitakin, voidaan yhdistää uusiksi vektoreiksi käyttäen yhdistämisfunktiota `c()`. Saatu vektoriarvoinen muuttuja voidaan tallentaa muuttujaan aivan kuten skalaaritkin. Itse asiassa R:ssä skalaaritkin ovat oikeasti yksipaikkaisia vektoreita.

Monesti yksinkertaisia aineistoja on helppo käsitellä vektorimuodossa, jolloin vektorin arvot olisivat esimerkiksi mittaustuloksia jostakin kokeesta.

Esimerkki 7. Luodaan vektorit $(0, 1, 2, 3)$ ja $(0, 1, 2, 3, 4, 5, 6)$:

```
> a<-c(0,1,2,3)
> b<-c(a,4,5,6)
> a
[1] 0 1 2 3
> b
[1] 0 1 2 3 4 5 6
```

Esimerkki 8. Edellisen esimerkin vektori saadaan myös seuraavilla tavoilla:

```
> 0:6
[1] 0 1 2 3 4 5 6
> seq(0,6,by=1)
[1] 0 1 2 3 4 5 6
```

Esimerkki 9. Nämä toimivat myös toiseen suuntaan ja erilaisilla väleillä:

```
> 6:0
[1] 6 5 4 3 2 1 0
> seq(0,100, by=10)
 [1]   0  10  20  30  40  50  60  70  80  90 100
> seq(0,6, by=0.5)
 [1] 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0
```

2.2 Vektorien käsittely

Vektorin alkioihin voidaan viitata antamalla halutut indeksit hakasuluissa. Huom. R:ssä indeksointi alkaa 1:stä eikä 0:sta. Jos viitataan indeksiin, jota ei ole vektorissa, tuloksena on puuttuva arvo, eli `NA`.

Esimerkki 10. Luodaan vektori `a` ja valitaan sen alkioita:

```
> a <- c(3,6,30,3,0)
> a[1]
[1] 3
> a[5]
[1] 0
> a[6]
[1] NA
```

Voidaan valita myös useita eri alkoita kerralla antamalla indeksiksi vektori.

Esimerkki 11. (Jatkoa edelliseen) valitaan vektorin `a` kolme viimeistä alkiota eri tavoilla:

```
> a[c(3,4,5)]
[1] 30  3  0
> a[3:5]
[1] 30  3  0
> a[3:length(a)]
[1] 30  3  0
> a[c(-1,-2)]
[1] 30  3  0
> a[c(F,F,T,T,T)]
[1] 30  3  0
```

Vektoriin voidaan valinnan kautta myös sijoittaa arvoja. Vektorin komponenttien arvoja voidaan vertailla käyttämällä hakasulkujen sisällä vektoria itseään vertailun kohteena. Tällöin saadaan totuusarvovektori, joka valitsee halutut komponetit.

Esimerkki 12. Luodaan vektori ja muutetaan kaikki kolmella jaolliset luvut nollaksi:

```
> a <- 1:10
> a%%3==0
 [1] FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE
> a[a%%3==0]<-0
> a
 [1]  1  2  0  4  5  0  7  8  0 10
```

Funktio `length()` palauttaa vektorin pituuden. Negatiiviset indeksit taas palauttavat koko vektorin lukuunottamatta näitä indeksejä, esimerkiksi `a[c(-1,-2)]` antaa vektorin `a` lukuunottamatta sen ensimmäistä ja toista alkoita. Vektoria voidaan indeksoida loogisella vektorilla, jolloin valitaan alkiot, joiden kohdalla on indeksivektorin arvo on `TRUE`.

Edellisen esimerkin viimeisestä tapaa valita vektorin alkoita voidaan käyttää vektorin alkoiden valitsemiseen ehtolauseiden avulla, mikä tulee jatkossa olemaan erittäin kätevää osa-aineistojen valitsemisessa. Ehtolause `a > 3` vertaa jokaista `a`:n alkiota lukuun kolme ja palauttaa vertailun tuloksen totuusarvovektorina:

Esimerkki 13.

```
> a > 3
```

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

Nyt käytettäessä ehtolausetta `a > 3` `a`:n indeksinä, sen pitäisi palauttaa `a`:n toinen ja kolmas alkio, eli juuri ne `a`:n alkiot, jotka ovat suurempia kuin 3.

Esimerkki 14.

```
> a[a > 3]
[1]  6 30
```

Juuri kuten halusimmekin. Voimme siis kätevästi valita alkoita vektorista ehtolauseiden avulla.

2.3 Matriisien luominen

R:ssä matriisi luodaan vektorista `matrix()` -komennolla. Funktio tarvitsee myös tiedon siitä, onko annettava data järjestetty riveittäin vai sarakkeittain (byrow).

Esimerkki 15. Luodaan matriisi

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

käyttäen komentoa `matrix()`

```
> matrix(c(1,2,3,4,5,6,7,8,9), ncol=3, nrow=3, byrow=TRUE)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

Edellisessä siis yhdistettiin luvut 1-9 komennolla c() vektoriksi ja kutsuttiin komentoa matrix(). Argumentteina funktiolle matrix() annettiin matriisin dimensiot 3x3 ja tieto, että data on järjestetty riveittäin. TRUE voidaan myös lyhentää T, kuten jatkossa tehdään. Kokeile! Vastaavasti argumentin arvoksi voidaan antaa FALSE tai F. Riveittäin ja sarakkeittain järjestämisen eroa voit kokeilla antamalla dataksi c(1,4,7,2,5,8,3,6,9).

2.4 Matriisien käsittely

Matriiseja voidaan käsitellä kuten vektoreitakin: hakasulkuja käyttämällä voidaan valita alkioita ja tehdä vertailuja. Matriiseista voidaan yksittäisten solujen lisäksi valita myös kokonaisia rivejä tai sarakkeita.

Esimerkki 16. Solujen, rivien ja sarakkeiden valitseminen:

```
> a <- matrix(c(1,2,3,4,5,6,7,8,9), ncol=3, nrow=3, byrow=TRUE)
> a[,1]
[1] 1 4 7
> a[1,]
[1] 1 2 3
> a[1,2]
```

```
[1] 2
> a[a>5]
[1] 7 8 6 9
```

Esimerkki 17. Arvojen sijoittaminen matriisiin:

```
> a <- matrix(c(1,2,3,4,5,6,7,8,9), ncol=3, nrow=3, byrow=TRUE)
> a[2,1]<-9001
> a
     [,1] [,2] [,3]
[1,]    1    2    3
[2,] 9001    5    6
[3,]    7    8    9
> a[,1] <- c(1,2,3)
> a
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    2    5    6
[3,]    3    8    9
```

Matriisiin voidaan lisätä rivejä ja sarakkeita käyttämällä komentoa `rbind()` ja `cbind`

Esimerkki 18. Rivien ja sarakkaiden lisääminen

```
> a <- matrix(c(1,2,3,4,5,6,7,8,9), ncol=3, nrow=3, byrow=TRUE)
> rbind(a, c(1,1,1))
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]    1    1    1
> cbind(rbind(a, c(1,1,1)), c(2,2,2,2))
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    2
[2,]    4    5    6    2
[3,]    7    8    9    2
[4,]    1    1    1    2
```

Rivien ja sarakkeiden poisto käy helpoiten käyttämällä totuusarvoja vektorioperaatioiden tapaan.

Esimerkki 19. Rivien ja sarakkaiden poistaminen

```
> a <- matrix(c(1,2,3,4,5,6,7,8,9), ncol=3, nrow=3, byrow=TRUE)
> a[,c(F,T,T)]
     [,1] [,2]
[1,]    2    3
[2,]    5    6
[3,]    8    9
> a[c(T,T,F),]
```

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

2.5 Vektorien ja matriisien laskutoimitukset

Matriiseille saadaan laskettua matriisien tulo operaattorin %*% avulla. Pelkkä kertomerkki tuottaa tässä hieman erilaisen tuloksen, kokeile! Matriisien yhteen- ja vähennyslaskut toimivat tavallisesti käyttäen operaattoreita + ja -. Transpoosi saadaan komennolla t(), jolle argumentiksi annetaan transponoitava matriisi.

Esimerkki 20. Merkitään

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

ja

$$B := \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}.$$

Lasketaan matriisitulo AB:

```
> A <- matrix(c(1,2,3,4,5,6,7,8,9), ncol=3, nrow=3, byrow=T)
> B <- matrix(c(9,8,7,6,5,4,3,2,1), ncol=3, nrow=3, byrow=T)
> A%*%B
     [,1] [,2] [,3]
[1,]   30   24   18
[2,]   84   69   54
[3,]  138  114   90
```

Esimerkki 21. Transponoidaan edellisen esimerkin matriisi A:

```
> t(A)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Esimerkki 22. Vektorien ja matriisien tulo onnistuu täysin samalla tavalla:

```
> a<-c(0,1,2,3)
> a %*% diag(4)
     [,1] [,2] [,3] [,4]
[1,]    0    1    2    3
```

Esimerkki 23. Matriisin käänteismatriisi saadaan käyttämällä funktiota `solve()`:

```
> A <- matrix(as.integer(runif(n = 4, min=1, max=20)), nrow=2)
> A
     [,1] [,2]
[1,]   15    6
[2,]    5    4
> solve(A)
           [,1] [,2]
[1,]  0.1333333 -0.2
[2,] -0.1666667  0.5
```

3 Tehtäviä

3.1 Tehtäviä

1. a) Tallenna laskun $4 + 10$ tulos muuttujaan `a`.
   b) Kerro `a` viidellä ja tallenna tulos takaisin muuttujaan `a`.
   c) Korota `a` kolmanteen potenssiin ja tallenna tulos muuttujaan `b`.
   d) Laske $e^{\texttt{a}}$ ja tallenna tulos muuttujaan `c`. Vihje: funktio `exp()`.
   e) Ota 10:s juuri `c`:stä ja tallenna tulos takaisin muuttujaan `c`.
   f) Laske sin(`b`) ja tallenna tulos muuttujaan `d`
   g) Tulosta summa `a+b+c+d`

2. Mitä tekee operaattori %%? Vihje: Laske esimerkiksi 51%%7 ja päättele lopputuloksesta.

3. Mitä tekee operaattori %/%? Vihje: Laske 51%%7 + 51%/ %7*7 ja päättele lopputuloksesta.

4. a) Luo vektori $(20, 5, -2, 3, 47)$ ja tallenna tulos vektoriin `v1`.
   b) Luo vektori, joka sisältää alkiot nollasta sataan viiden välein, eli vektori $(0, 5, 10, \cdots, 100)$ ja tallenna tulos muuttujaan `v2`.
   c) Yhdistä vektorit `v1` ja `v2` vektoriksi `v3`.
   d) Valitse vektorista `v3` ne alkiot, jotka ovat suurempia kuin 3 ja pienempiä kuin 50 ja tallenna tulos vektoriin `v4`.
   e) Valitse vektorista `v4` ne alkiot, jotka ovat jaollisia kymmenellä.

5. Luo vektori komennolla `a <- rep(0,1,50)`.
   a) Valitse nyt vektorin `a` parilliset indeksit ja muuta niiden arvoksi 2. Vihje: Kokeile ensin vektorin `b<-1:50` kanssa, mitä valinta `b[c(F,T)]` tekee.
   b) Laske vektorin komponenttien summa käyttämällä funktiota `sum()` ja tulosta se.
   c) Valitse nyt parittomat indeksit ja anna niille sellainen arvo, että saat vektorin `a` komponenttien summaksi 80.

6. Merkitään

$$A := \begin{bmatrix} 3 & 5 & 6 \\ \frac{1}{2} & \sqrt{5} & 16 \\ 0 & 2 & 0 \end{bmatrix}.$$

   Luo matriisi A R:ään. *Vihje: neliöjuuren saat komennolla sqrt(). R antaa solun sisällön desimaalilukuna.*

7. Laske edellisen tehtävän matriisin käänteismatriisi, kerro se *pystyvektorilla* $(1, 1, 0)$ ja tallenna matriisiksi B.

8. Luo 3x3 identiteettimatriisi I_3 ja testaa toimiiko $AI = A$, kun käytät matriisikertolaskuun operaattoria `%*%`? Entä jos vain `*`? Vihje: Aiemmin esimerkissä esiintynyt diag() funktio.

9. Onko edellisten tehtävien matriisi A symmetrinen? Entä identiteettimatriisi I_3? Vihje: Transpoosi ja operaattori ==.

10. Muodosta matriisi käyttämällä komentoa `C <- matrix(c(runif(20, min = 1, max=20)), ncol=4)`.

    a) Kuinka monen solun arvo on pienempi kuin 5? Vihje: Esimerkki 16 ja funktio `length()`.

    b) Muuta nyt jokainen viittä pienempi arvo olemaan tasan 10. Vihje: Käytä edellisen tehtävän tulosta ja sijoitusoperaattoria.

    c) Poista nyt matriisista `C` ensimmäinen sarake ja viimeinen rivi. Vihje: Esimerkki 19.

11. Luo nyt uusi matriisi komennolla `C <- matrix(1:100, ncol=2)`. Muuta jokaisen parillisen luvun tilalle puuttuva arvo, eli `NA`.

12. Jatka edellisen tehtävän matriisin kanssa ja muuta nyt jokainen puuttuva arvo luvuksi 0. Vihje: `is.na()`. Lisäpohdittavaksi: Mitä käy, jos vertailet mitä tahansa muuttujaa arvon `NA` kanssa? Esim. `a == NA`.