Johdantoa matriisien ja lineaaristen mallien käsittelyyn R-ohjelmistolla

Anne Oikarinen

Matematiikan, tilastotieteen ja filosofian laitos
Tampereen yliopisto

elokuu 2004

Sisältö

1 Yleistä R:stä **2**

2 R:n syntaksi ja työskentelyn aloittaminen **3**

2.1 Ohjeiden käyttö R-istunnon aikana 4

2.2 Lisäpaketit ja aineistot 4

3 Vektorit **6**

3.1 Vektorien muodostaminen 6

3.2 Vektorien käsittelystä ja laskutoimituksista 9

4 Matriisit **11**

4.1 Matriisien muodostaminen 11

4.2 Matriisien laskutoimitukset 15

4.3 Joitakin matriisikomentoja 16

5 Lineaarisista malleista ja grafiikasta **25**

5.1 Grafiikasta 27

5.2 Sekamalleista 29

6 Saatavilla olevaa R-materiaalia **36**

6.1 Jari Oksasen opas ekologeille 36

6.2 R-ohjelmiston kotisivujen sisällöstä 36

6.3 Muuta materiaalia 37

1 Yleistä R:stä

R on tilastollinen komentopohjainen ohjelmisto. Se on ilmaisjakeluohjelma, joten se on vapaasti ladattavissa, käytettävissä ja levitettävissä. R:n kirjoittivat alunperin Robert Gentleman ja Ross Ihaka Aucklandin yliopiston tilastotieteen laitokselta, Uudesta-Seelannista. Vuoden 1997 puolivälin jälkeen tietyllä ryhmällä on ollut oikeus muuttaa R:n lähdekoodia. Lisäksi suuri joukko yksityisiä ihmisiä on vaikuttanut R:n kehitykseen muun muassa suorittamalla virheraportointia ja dokumentointia. R kehittyy ohjelmana jatkuvasti. Uusin versio on kesäkuussa 2004 julkaistu versio 1.9.1. Komentopohjaisuus kauhistuttaa usein graafisiin, hiirtä klikkaamalla toimiviin ohjelmiin tottunutta käyttäjää, mutta R paljastuu yllättävän yksinkertaiseksi käyttää. Useat suomenkieliset oppaat ja lukematon määrä englanninkielisiä käyttöohjeita helpottavat alkuun pääsyä. R-ohjelma sisältää monia tapoja muodostaa ja käsitellä vektoreita ja matriiseja ja tässä paperissa perehdytäänkin nimenomaan tähän. Lisäksi tehdään pintapuolinen silmäys lineaaristen ja sekamallien muodostamiseen ja käsittelyyn, sekä esitellään tarkemmin muutamia internetin R-sivustoja. Tämän työn tarkoituksena ei ole muodostaa perusteellista opasta R:n käytöstä aloittelijalle. Tärkein päämäärä on opettaa hallitsemaan vektoreiden ja matriisien muodostaminen ja käsittely ja antaa ohjeita siitä mistä ja miten apua kannattaa etsiä ongelmatilanteissa.

2 R:n syntaksi ja työskentelyn aloittaminen

Muutaman yksinkertaisen komennon opettelemisen jälkeen R:n käyttö on suhteellisen suoraviivaista ja helppoa. Laskutoimitukset kirjoitetaan tietyn syntaksin mukaan ruudulle, joka on pitkälle samanlainen kuin normaalisti paperilla laskettaessa. Tämän jälkeen enteriä painamalla saadaan tulokset ja/tai mahdolliset virheilmoitukset. R:ssä on paljon valmiita funktioita, joita voi käyttää tietyn komennon avulla (itse funktion muotoa tahi varsinaista toimintapaa ei usein ole tarpeen tuntea). Esimerkkeinä tällaisista komennoista voisi olla *sqrt()* neliöjuuren laskemista varten, tai *mean()*, joka laskee keskiarvon, *sd()* keskihajontaa varten ja niin edelleen. Valmiit funktiot nopeuttavat huomattavasti työskentelyä joskin niitä on mahdollista kirjoittaa myös itse. Ohjelmointia harjoittaneelle R:n funktioiden muodostaminen ei tuottane suurempia vaikeuksia, eikä muidenkaan sitä ole ylivoimaisen vaikeaa oppia.

R on hyvin tarkka kirjoitusasusta. R:ssä pienten ja isojen kirjainten välillä on suuri ero. Esimerkiksi R ei tunnista funktiota jos se on kirjoitettu isolla alkukirjaimella. Yleensä R:llä työskenneltäessä on tarpeen nimetä lasketut tulokset tai muodostettavat vektorit ja matriisit. Näin näitä päästään myöhemmin käsittelemään vain viittaamalla esimerkiksi juuri matriisin nimeen, eikä matriisia tarvitse muodostaa jokaista laskua varten erikseen. Nimen antaminen tulokselle tai matriisille tapahtuu sijoittamalla. Sijoittaminen tehdään kirjoittamalla ensin haluttu nimi sen jälkeen peräkkäin "<" pienempi kuin ja "-" eli tavuviiva ja sen jälkeen mitä kyseiseen nimeen halutaan sijoittaa. Sijoittaminen voidaan tehdä myös toiseen suuntaan, eli ensin kirjoitetaan tulos sen jälkeen "nuoli" oikealle ja lopuksi haluttu nimi. Esimerkiksi

```
> v<-c(1,2,0)
> v
[1] 1 2 0
```

sijoittaa v:hen vektorin, jonka alkiot ovat 1, 2 ja 0. Katso tarkemmin vektorien muodostamisesta luku 3.

```
> c(9,0,-1)->vek
> vek
```

```
[1]  9  0 -1
```

2.1 Ohjeiden käyttö R-istunnon aikana

Kirjoittamalla näytölle *?komento* tai *help(komento)* saa tietoa näitten komentojen (funktioiden) toiminnasta. Kuvaukset ovat usein kuitenkin lyhyitä ja voivat edistyneemmästäkin R:n käyttäjästä tuntua epäselviltä. Yleensä lopuksi annetaan esimerkki komennon käytöstä, joka usein valaisee tilannetta tehokkaimmin. Helppien käytössä on se ongelma, että komennon nimi täytyy tietää voidakseen etsiä sitä *help*:n tai *?*:n avulla. Jos tätä nimeä ei tiedä voi mahdollista komentoa yrittää etsiä kirjoittamalla *help.search("oletettu komento")*. Vasta-alkajalle eniten hyötyä lieneekin *html*-helpistä, josta voi etsiä hakusanan avulla komentoja tahi selata suoraan eri pakkausten (ks. luku 2.2) komentoja läpi. Vaikka funktion nimen tietäisikin on usein paikallaan vilkaista helpistä esimerkiksi annettavien parametrien muoto ja määrä ja varmistua että komento todellakin laskee haluttua asiaa, eikä vain jotain sinnepäin.

2.2 Lisäpaketit ja aineistot

Avatessaan R:n saa käyttöönsä automaattisesti paketin nimeltä *base*. Jo tämä paketti sisältää kaikkein keskeisimmät menetelmät ja funktiot. Suuri osa R:n arvokkaista ohjelmista on kuitenkin erillisissä lisäpaketeissa (*packages*) eli kirjastoissa (*libraries*). Osa näistä paketeista tulee perus-R:n mukana, mutta osa on erikseen hankittava ja asennettava. Ne ovat saatavilla samasta paikasta kuin R (http://cran.r-project.org/).

Vaikka kirjasto olisi asennettu, ei R näe sen ohjelmia, ennen kuin kirjasto on otettu käyttöön. Seuraavat komennot ovat hyödyllisiä kirjastoja käytettäessä:

```
> library()            # Luettelo asennetuista kirjastoista
> library(help=nlme)   # Kirjaston nlme ohjelmat
> library(nlme)        # Ottaa käyttöön kirjaston nlme
```

```
> detach(package:nlme) # Poistaa käytöstä kirjaston nlme
```

Nämä komennot ja kuvaus pakkauksista pohjautuvat Jari Oksasen teokseen R: opas ekologeille (2003). Tämä opas löytyy netistä osoitteesta http://cc.oulu.fi/∼jarioksa/opetus/.

R:ssä on valmiina suuri joukko esimerkkiaineistoja. Listan aineistoista saa kirjoittamalla

```
> data()
```

Tarkemman kuvauksen kustakin aineistosta, muun muassa aineiston muuttujista, saa komennoilla *?aineiston nimi* tai *help(aineiston nimi)*. Aineiston saa käyttöönsä komennolla *data(aineiston nimi)*. Tämän jälkeen aineiston muuttujiin pääsee käsiksi kirjoittamalla *aineiston nimi$muuttujan nimi*. Tämä kuitenkin vaikuttaa turhan isotöiseltä. Helpompaa on liittää aineisto nykyiseen istuntoon *attach(aineiston nimi)*-komennolla, jonka jälkeen voimme viitata suoraan aineiston muuttujien nimiin. Komennolla *names(aineiston nimi)* saadaan kätevästi aineiston muuttujien nimet näkyviin. Komennolla *detach(aineiston nimi)* aineisto puolestaan poistetaan käytöstä. Käyttämättömät aineistot on syytä todellakin poistaa käytöstä, ettei samannimisten muuttujien kanssa tule sekaannuksia.

3 Vektorit

3.1 Vektorien muodostaminen

Vektorin muodostaminen R-ohjelmistolla on varsin yksinkertaista. Tämä on mahdollista useammalla eri tavalla, mutta kaikista yksinkertaisin keino lienee luetella vektorin alkiot ja yhdistää nämä komennolla c (*concatenate*), joka oletusarvoisesti yhdistää alkiot vektorin muotoon. Siis sijoitusoperaatiolla

```
> vek1<-c(1,8,3.0,435)
> vek1
[1]   1   8   3 435
```

Toinen keino muodostaa vektori on *array*-komennolla.

```
> vek2<-array(c(0,0.5,7,1),dim=c(4))
> vek2
[1] 0.0 0.5 7.0 1.0
```

Tässä vektorin dimensio voidaan antaa erillisellä parametrillä. Vektorin tapauksessa on vain yksi dimensio, joka tässä tapauksessa on 4. Dimensiota muuttamalla voidaan aineistosta muodostaa eri pituisia vektoreita. R:n komennot ovat varsin joustavia ja kuten tässäkin tapauksessa huomataan, komentoa vektorin muodostamiseksi voidaan entisestään lyhentää. Esimerkiksi

```
> vek3<-array(c(7,1),5)
> vek3
[1] 7 1 7 1 7
```

Tässä tapauksessa vektoria jatketaan toistamalla dataa niin kauan kunnes annettu dimensio on täynnä. Peräkkäisistä arvoista koostuva vektori voidaan muodostaa helposti kaksoipisteen ":" avulla.

```
> x1<-3:1
> x1
[1] 3 2 1
```

```
> y1<--2.1:3
> y1
[1] -2.1 -1.1 -0.1  0.9  1.9  2.9
```

Näin on mahdollista kuitenkin muodostaa vain peräkkäisistä arvoista (yhden yksikön välein) koostuvia vektoreita. Jos sarja halutaan tehdä jonkun muun säännöllisen välin mukaan täytyy käyttää *seq*-komentoa.

```
> seq(-1,0.5,by=0.3)
[1] -1.0 -0.7 -0.4 -0.1  0.2  0.5
> seq(-1,0.5,length=5)
[1] -1.000 -0.625 -0.250  0.125  0.500
```

Funktion kaksi ensimmäistä parametria kertovat sarjan alku- ja loppupisteet. *by*-parametri kertoo minkä kokoisin välein sarja halutaan muodostaa. *length*-komento puolestaan kertoo kuinka monta alkiota sarjaan halutaan, jolloin R automaattisesti jakaa annetun välin sopivan mittaisiin osiin. Toinen usein kätevä komento on *rep*, joka toistaa annettua arvoa tai arvoja halutun määrän.

```
> rep(c(-1,0),3)
[1] -1  0 -1  0 -1  0
```

Tässä siis toistetaan -1 ja 0 koostuvaa vektoria, viimeisen paremetrin ilmoittaman lukumäärän (3) verran. Tietenkin on mahdollista muodostaa muista kuin numeroista koostuvia vektoreita. Merkkijonot on kirjoitteva tällöin lainausmerkkien sisään.

```
> sanoja<-c("a","Tampere","jotain")
> sanoja
[1] "a"       "Tampere" "jotain"
```

Pituus ja tyyppi

Vektoreilla on aina pituus ja tyyppi ja kaikkien vektorin alkioiden on oltava samaa tyyppiä. Pituus voidaan tarkistaa komennolla *length* ja tyyppi komennolla *mode*. Esimerkiksi

```
> length(sanoja)
[1] 3
> mode(sanoja)
[1] "character"

> mode(vek1)
[1] "numeric"
```

numeric luonnollisesti viittaa numeeriseen muuttujaan ja *character* merkkijono tyyppiseen muuttujaan. Kolmas tavallinen tyyppi on looginen muuttuja (logical), joka voi saada vain arvoja tosi (TRUE) tai epätosi (FALSE). Looginen muuttuja syntyy yleensä ehtolauseiden tuloksena.

```
> vek1
[1]   1   8   3 435
> loogi<-vek1>3
> loogi
[1] FALSE  TRUE FALSE  TRUE
```

Loogisia muuttujia joutuu käyttämään kun kutsuttavalle funktiolle on kerrottava, onko tietyn parametrin arvo tosi vai epätosi; tällöin ne saa lyhentää muotoon T ja F (Oksanen 2003). R:ssä myös merkkijonomuuttujia voi käyttää luokiteltavina muuttujina analyyseissä. Tällöin niitä kutsutaan faktoreiksi. Sekä numeerisen että merkkijonomuuttujan voi muuntaa faktoriksi komennolla *factor*.

```
> factor(sanoja)
[1] a       Tampere jotain
Levels: a jotain Tampere
```

On tärkää huomata, että muutettaessa numeerinen muuttuja faktoriksi, ei muuttujan arvoja voida enää käyttää numeroiden tavoin. Faktoriksi muuntamisen jälkeen numeroista tulee luokkien ”nimiä”.

```
> vek7
 [1]  1  1  1  2  7  9 -3  2  2 -3
> fakvek7<-factor(vek7)
> fakvek7
 [1] 1  1  1  2  7  9  -3 2  2  -3
Levels: -3 1 2 7 9
> mean(fakvek7)
[1] NA
Warning message:
argument is not numeric or logical: returning NA in: mean.default(fakvek7)
```

Yritettäessä laskea keskiarvoa saadaan siis tulokseksi varoitus ja ”NA”, joka viittaa puuttuvaan tietoon.

3.2 Vektorien käsittelystä ja laskutoimituksista

Vektorin alkioihin voidaan viitata komennoilla $nimi[i]$, missä *nimi* on vektorin nimi ja i on alkion indeksi johon halutaan viitata. Esimerkiksi edellisessä luvussa muodostetulle vek1 nimiselle vektorille $vek1[2]$ poimii vektorista toisen alkion eli numeron 8.

```
> vek1[2]
[1] 8
```

Viittaaminen voidaan lonnollisestikin tehdä useampiin alkioihin kerrallaan. Tästä on suurta hyötyä kun halutaan käsitellä suurista vektoreista vain osia. Seuraavista komennoista ensimmäinen poimii alkiot $2-4$ ja jälkimmäinen vain luetellut alkiot, eli alkiot paikoilta 2 ja 4.

```
> vek1[2:4]
[1]   8   3 435
> vek1[c(2,4)]
[1]   8 435
```

Loogisia operaattoreita käytetään usein valitsemaan osa-aineistoja: kun vektorin indeksinä on looginen vektori, valitaan ne alkiot, joille ehto on tosi (Oksanen 2003). Esimerkiksi voimme valita pienemmät ja yhtäsuuret arvot kuin 1 kirjoittamalla

```
> vek7[vek7<=1]
[1]  1  1  1 -3 -3
```

Hakasuluissa olevan ehtolauseen tulos on vektori, joka on TRUE alkioille 1, 2, 3, 7 ja 10. Yllä käytettiin vertailuoperaattoria <=. Muita vertailuoperaattoreita <, >, =>, == (yhtäläisyys), ! = (epäyhtäläisyys) ja ! (negaatio).

Vektoreille on luonnollisesti voimassa kaikki peruslaskutoimitukset eli yhteen-, vähennys-, kerto-, ja jakolaskut, jotka suoritetaan alkioittain. Esimerkiksi siis komento

```
> vek1-7
```

tuottaa tulokseksi vektorin, jossa kustakin vek1-vektorin alkiosta on vähennetty luku 7 eli

```
[1]  -6   1  -4 428
```

Nyt on tärkeää huomata, että kun molemmat laskutoimitusten alkiot ovat vektoreita, suoritetaan laskut silti alkioittain, siis esimerkiksi kertolasku

```
> vek1*vek2
[1]   0   4  21 435
```

kertoo kunkin alkion keskenään.

Vektorikertolasku

Vektorikertolasku määritellään komennolla %*%. Esimerkiksi edellä määritellyille vektoreille

```
> vek1
[1]   1   8   3 435
> vek2
[1] 0.0 0.5 7.0 1.0
> vek1%*%vek2
     [,1]
[1,]  460
```

Huomaa, että vaikka emme erikseen ”transponoineet” toista vektoreista, osaa R silti laskea tuloksen oikein, kunhan vektorit vain ovat sopivan mittaisia.

Transpoosi

Vektorin transpoosi saadaa komennolla ”*t()*”. Transpoosi (*transpose*) kohtelee vektoria pystyvektorina. Transponoinnin jälkeen tulokseksi saadaan itseasissa $1 \times p$ matriisi.

```
> t(vek2)
     [,1] [,2] [,3] [,4]
[1,]    0  0.5    7    1
> t(t(vek2))
     [,1]
[1,]  0.0
[2,]  0.5
[3,]  7.0
[4,]  1.0
```

4 Matriisit

4.1 Matriisien muodostaminen

Matrisin muodostaminen voidaan hoitaa usealla eri tavalla. Esimerkiksi vektorien muodostamisessa käytettyä *array*-komentoa käyttämällä. Nyt dimensioiksi täytyy vain antaa kaksi parametria vastaamaan rivien ja sarakkeiden lukumäärää. Esimerkiksi

```
> mat1<-array(c(1:6),dim=c(2,3))
> mat1
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Huomaa, että matriisi täytetään sarakkeittain, vrt.

```
> mat2<-array(c(1:6),dim=c(3,2))
> mat2
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

Matriisin muodostamista varten on R:llä olemassa myös oma *matrix*-komento. Yksinkertaisimmillaan komento toimii seuraavasti

```
> mat3<-matrix(12:1,nrow=4,ncol=3)
> mat3
     [,1] [,2] [,3]
[1,]   12    8    4
[2,]   11    7    3
[3,]   10    6    2
[4,]    9    5    1
```

Tässä siis parametreinä annetaan matriisin alkiot, matriisin rivien lukumäärä (nrow tai nr *number of rows*) ja matriisin sarakkeiden lukumäärä (ncol tai nc *number of columns*). R-ohjelma ei välttämättä vaadi sekä sarakkeiden, että rivien lukumäärän ilmoittamista. Esimerkiksi

```
> mat4<-matrix(12:1,nrow=3)
> mat4
     [,1] [,2] [,3] [,4]
[1,]   12    9    6    3
[2,]   11    8    5    2
[3,]   10    7    4    1
```

R siis osaa aineiston koon perusteella päätellä tarvittavien sarakkeiden (tai rivien) lukumäärän, kunhan vain alkioiden määrä jakautuu tasaisesti riveille ja sarakkeille. Usein on tarpeen muodostaa pelkästään yhdestä numerosta muodostuva matriisi. Tällä komennolla se on helppoa.

```
> mat5<-matrix(1,nrow=2,ncol=3)
> mat5
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    1
```

Matriisin kaikkia alkioita ei siis ole tarve luetella. Tämä helpottaa huomattavasti suurien matriisien muodostamista. Tälläkin komennolla matriisi täytetään oletusarvoisesti sarakkeittain. *Matrix*-komentoon voidaan kuitenkin liittää *byrow*-komento ja asettamalla tämän komennon arvo "T":ksi (true) saadaan matriisi täytettyä riveittäin. Siis

```
> mat6<-matrix(c(1,3,5,7,9,11),nrow=2,ncol=3,byrow=T)
> mat6
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    7    9   11
```

Matriiseja voitaisiin muodostaa myös yhdistelemällä vektoreita. R:llä on käytössä *rbind*- ja *cbind*-komennot, joilla matriiseja tai vektoreita voidaan yhdistää riveittäin tai sarakkeittain. Muodostetaan ensin kolme vektoria, jonka jälkeen yhdistetään näitä sekä sarakkeittain että riveittäin.

```
> vek4<-c(1,3)
> vek5<-c(2,1.9)
> vek6<-c(0,20)
```

```
> m1<-cbind(vek4,vek5,vek6)
> m1
     vek4 vek5 vek6
[1,]    1  2.0    0
[2,]    3  1.9   20
> m2<-rbind(vek4,vek5,vek6)
> m2
     [,1] [,2]
vek4    1  3.0
vek5    2  1.9
vek6    0 20.0
```

R:n komennot ovat siitä joustavia, että monella komennolla saadaan aikaan vektoreita tahi matriiseja sen mukaan mitkä dimensiot komentoon syötetään. R:llä on myös oma komentonsa *diag*() esimerkiksi diagonaalimatriisien muodostamiseksi, toki tämäkin voidaan tehdä myös yllä mainittujen komentojen avulla. Nyt siis esimerkiksi

```
> diag(4)
```

tuottaa 4×4 identiteettimatriisin. Tällä samalla komennolla voidaan muodostaa myös muunlaisia diagonaalimatriiseja, esimerkiksi

```
> dmat<-diag(c(1,2,3))
> dmat
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    2    0
[3,]    0    0    3
```

R:n komentoja voidaan usein myös yhdistää toisiinsa, esimerkiksi

```
> diag(3,nr=2,nc=4)
     [,1] [,2] [,3] [,4]
[1,]    3    0    0    0
[2,]    0    3    0    0
```

Tässä siis rivien ja sarakkeiden lukumäärän ilmoittavat komennot on yhdistetty diagonaalimatriisin muodostamiseen.

Matriisin dimensiot

Matriisin dimensiot saadaan halutessa tarkistettua komennolla *dim*. Tämä on hyödyllinen komento varsinkin käsiteltäessä suuria aineistoja, joissa matriiseissa voi olla jopa kymmeniä tuhansia rivejä.

```
> mat4
     [,1] [,2] [,3] [,4]
[1,]   12    9    6    3
[2,]   11    8    5    2
[3,]   10    7    4    1
> dim(mat4)
[1] 3 4
```

eli tässä matriisissa on kolme riviä ja neljä saraketta.

Alkioon, riviin tai sarakkeeseen viittaaminen

Muodostetaan ensin viidestä ensimmäisestä kokonaisluvusta ja niiden neliöistä koostuva matrisi. Huomaa, että molemmilla sarakkeilla on nyt nimet: "luvut"ja "neliöt".

```
> luvut<-c(0,1,2,3,4)
> neliöt<-c(luvut)^2
> mat7<-cbind(luvut,neliöt)
> mat7
     luvut neliöt
[1,]     0      0
[2,]     1      1
[3,]     2      4
[4,]     3      9
[5,]     4     16
```

Matriisin alkioihin viittaaminen tapahtuu samoin kuin vektoreiden tapauksessakin, paitsi että nyt annetaan kaksi dimensiota vastaamaan rivejä ja sarakkeita.

```
> mat7[4,2]
[1] 9
```

eli ensimmäinen parametri viittaa riveihin ja toinen sarakkeisiin. Tässä siis viitattiin matriisin mat7, neljännen rivin toiseen sarakkeeseen. Luonnollisesti voidaan viitata myös kokonaiseen riviin tai sarakkeeseen, tämä tapahtuu jättämällä toisen parametrin paikka tyhjäksi.

```
> mat7[3,]
 luvut neliöt
     2      4
> mat7[,2]
[1]  0  1  4  9 16
```

Nyt nimettyihin sarakkeisiin voitaisiin viitata myös sarakkeen nimen avulla, esimerkiksi

```
> mat7[,"neliöt"]
[1]  0  1  4  9 16
```

4.2 Matriisien laskutoimitukset

Matriisikertolasku

Matriiseille ovat niin ikään voimassa normaalit laskutoimitukset, nämä suoritetaan aina alkioittain. Varsinainen matriisikertolasku suoritetaan samalla komennolla kuin vektoreillekin eli merkkiyhdistelmällä %*%. Esimerkiksi

```
> m1
     vek4 vek5 vek6
[1,]    1  2.0    0
[2,]    3  1.9   20
> m2
     [,1] [,2]
vek4    1  3.0
vek5    2  1.9
vek6    0 20.0
> m3<-m1%*%m2
> m3
```

```
      [,1]   [,2]
[1,]  5.0   6.80
[2,]  6.8 412.61
```

4.3 Joitakin matriisikomentoja

Monia laskutoimituksia varten on R:ssä olemassa valmiita komentoja, mutta näiden komentojen löytäminen pelkästään helppejä selailemalla voi olla vaikeaa. Tässä esitellään niistä joitakin

diag-komento
Edellä jo mainittua *diag*-komentoa voidaan käyttää myös matriisin diagonaalialkioiden poimimiseen. Esimerkiksi

```
> diag(m3)
```

tuottaa tulokseksi

```
[1]   5.00 412.61
```

transpoosi
Matriisin transpoosi saadaan yksinkertaisesti komennolla ”t”.

```
> m1
     vek4 vek5 vek6
[1,]    1  2.0    0
[2,]    3  1.9   20
> t(m1)
     [,1] [,2]
vek4    1  3.0
vek5    2  1.9
vek6    0 20.0
```

outer-komento

R:llä on mahdollista laskea myös muunlaisia matriisituloksia *outer*-komennon avulla. Käytetään tuttuja vek4 ja vek6 vektoreita näihin laskuihin

```
> vek4
[1] 1 3
> vek6
[1]  0 20
```

Nyt

```
> outer(vek4,vek6,"+")
     [,1] [,2]
[1,]    1   21
[2,]    3   23
> outer(vek4,vek6,"-")
     [,1] [,2]
[1,]    1  -19
[2,]    3  -17
> outer(vek4,vek6,"*")
     [,1] [,2]
[1,]    0   20
[2,]    0   60
> outer(vek4,vek6,"/")
     [,1] [,2]
[1,]  Inf 0.05
[2,]  Inf 0.15
```

Ensimmäinen sarake muodostuu soveltamalla vek4:n alkioihin lainausmerkeissä annetulla operaatiolla vek6:n ensimmäistä komponenttia. Toinen sarake muodostuu vastaavasti soveltamalla vek6:n toista alkiota. Esimerkiksi siis vähennyslaskulla saadaan ratkaisun toiseen sarakkeeseen $1-20=-19$ ja toiseksi alkioksi $3-20=-17$. Tätä samaa komentoa voidaan luonnollisesti soveltaa myös matriiseille. Esimerkiksi seuraaville matriiseille **a** ja **b**

```
> a<-mat1[1:2,1:2]
> a
```

```
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> b<-mat3[1:2,1:2]
> b
     [,1] [,2]
[1,]   12    8
[2,]   11    7
```

saadaan komennolla

```
> outer(a,b,"+")
```

neljä tulosmatriisia, joissa kussakin lisätään matriisiin a vuorollaan kukin b:n alkio.

```
, , 1, 1

     [,1] [,2]
[1,]   13   15
[2,]   14   16

, , 2, 1

     [,1] [,2]
[1,]   12   14
[2,]   13   15

, , 1, 2

     [,1] [,2]
[1,]    9   11
[2,]   10   12

, , 2, 2

     [,1] [,2]
```

```
[1,]    8   10
[2,]    9   11
```

Ensimmäinen matriisi saadaan lisäämällä a:n alkioihin, matriisin b ensimmäisen rivin ja ensimmäisen sarakkeen alkio 12. Toinen matriisi saadaan lisäämällä kuhunkin a:n alkioon luku 11, eli b-matriisin toisen rivin, ensimmäisen sarakkeen alkio. Kolmas matriisi muodostuu vastaavasti lisäämällä kuhunkin a:n alkioon luku 8 ja niin edelleen. Toinen esimerkki *outer*-komennon käytöstä

```
> outer(1:4,1:3,"^")
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    2    4    8
[3,]    3    9   27
[4,]    4   16   64
```

missä siis numeroita 1-4 korotetaan ensimmäiseen, toiseen ja kolmanteen potenssiin.

crossprod, ristitulo

Esimerkiksi komennolla *crossprod(x,y)* saadaan laskettua matriisitulo $\mathbf{x}'$:n ja $\mathbf{y}$:n väliltä. Esimerkissä käytetään jo tuttuja matriiseja a ja b.

```
> crossprod(a,b)
     [,1] [,2]
[1,]   34   22
[2,]   80   52
> t(a)%*%b
     [,1] [,2]
[1,]   34   22
[2,]   80   52
```

Yhtälöryhmän ratkaisu ja käänteismatriisit

R:llä voidaan myös ratkaista yhtälöryhmiä. Tämä tapahtuu *solve* komennon avulla. Tämä funktio ratkaisee x:n yhtälöstä

$$a * x = b,$$

missä b voi olla joko vektori tai matriisi. Esimerkiksi

```
> a
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> b
     [,1] [,2]
[1,]   12    8
[2,]   11    7
> x<-solve(a,b)
> x
     [,1] [,2]
[1,] -7.5 -5.5
[2,]  6.5  4.5
```

Nyt nähdään helposti, että a%*%x=b. Tämän saman komennon avulla onnistuu myös matriisin kääntäminen. Sillä jos parametriä b ei erikseen määritellä, asettaa R tämän paikalle identiteettimatriisin, jolloin ratkaisuna saadaan a:n käänteismatriisi. Esimerkiksi

```
> solve(a)
     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5
```

PNS-estimaatit

Pienimmän neliösumman ratkaisu saadaan joko yksinkertaisesti laskemalla lauseke $\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$ tai käyttämällä R:n valmista funktioita *lsfit*, joka siis laskee $\boldsymbol{\beta}$:n pienimmän neliösumman estimaatin mallissa $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \epsilon$. Komento *lsfit* saa parametreinään matriisin $\mathbf{X}$, jonka rivit vastaavat havaintoja ja sarakkeet muuttujia. Toinen parametri on vastemuuttuja $\mathbf{y}$. Lisäksi voidaan erikseen määritellä halutaanko vakiotermiä käyttää, oletusarvoisesti R ottaa tämän termin mukaan. Seuraavassa on kaksi esimerkkiä PNS-estimaattien laskemisesta sekä vakiotermin kanssa, että ilman sitä.

```
> x2<-0.5^(outer(1:3,1:3))
> x2
```

```
      [,1]     [,2]        [,3]
[1,] 0.500 0.250000 0.125000000
[2,] 0.250 0.062500 0.015625000
[3,] 0.125 0.015625 0.001953125
> y2
[1] 3 2 1
> PNS1<-solve(t(x2)%*%x2)%*%crossprod(x2,y2)
> PNS1
           [,1]
[1,]   7.333333
[2,]   8.000000
[3,] -21.333333

> lsfit(x2,y2,intercept=F)
$coefficients
        X1         X2         X3
  7.333333   8.000000 -21.333333

$residuals
[1] 0 0 0

$intercept
[1] FALSE ...
```

Vakiotermin kanssa

```
> x3<-outer(1:5,0:3,"^")
> x3
     [,1] [,2] [,3] [,4]
[1,]    1    1    1    1
[2,]    1    2    4    8
[3,]    1    3    9   27
[4,]    1    4   16   64
[5,]    1    5   25  125
> y3<-c(1,1,2,2,3)
> beta<-solve(t(x3)%*%x3)%*%crossprod(x3,y3)
```

```
> beta
           [,1]
[1,] 0.80000000
[2,] 0.07142857
[3,] 0.07142857
[4,] 0.00000000

> lsfit(x3,y3)
$coefficients
   Intercept           X1           X2           X3           X4
8.000000e-01 7.142857e-02 7.142857e-02 3.845688e-17 0.000000e+00

$residuals
[1]  0.05714286 -0.22857143  0.34285714 -0.22857143  0.05714286

$intercept
[1] TRUE ...
```

trace eli jälki

Matriisin jäljen, *trace*:n, laskemiseksi ei R:ssä ole valmista komentoa. Jälki saadaan kuitenkin laskettua yksinkertaisilla laskutoimituksilla

```
> sum(diag(a))
[1] 5
```

eli diagonaalialkioiden summana.

Ominaisarvot

Matriisien ominaisarvojen laskeminen tapahtuu komennolla *eigen*.

```
> m4<-matrix(c(1:3,2,1,4,3,4,1),byrow=T,nrow=3)
> m4
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    2    1    4
[3,]    3    4    1
```

```
> eigen(m4)
$values
[1]  7.074674 -0.886791 -3.187883

$vectors
           [,1]       [,2]       [,3]
[1,] -0.5057852  0.8240377 -0.2552315
[2,] -0.5843738 -0.5449251 -0.6013018
[3,] -0.6345775 -0.1549789  0.7571611
```

Oletusarvoisesti vastauksena saadaan siis sekä matriisin ominaisarvot että ominaisvektorit.

apply ja tapply

Eräillä komennoilla voidaan tehdä laskutoimituksia esimerkiksi kullekin rivillе ja sarakkeelle, tai tietyn "ominaisuuden mukaan" jaetuille vektoreille. Näitä komentoja on paljon *apply, tapply, sapply, lapply, mapply*. Mutta tässä esitellään vain yleisimmät ja kenties tarpeellisimmat eli *apply* ja *tapply*. Muodostetaan ensin keksityn aineiston pohjalta matriisi, johon on koottu tiedot kolmen henkilön laihduttamisen edistymisestä kolmen kuukauden ajalta.

```
> paino<-c(110,108,107.5,98,98.3,96,74,73,70)
[1] 110.0 108.0 107.5  98.0  98.3  96.0  74.0  73.0  70.0
> hlo<-c(rep(1,3),rep(2,3),rep(3,3))
> hlo
[1] 1 1 1 2 2 2 3 3 3
> aika<-c(rep(c(1,2,3),3))
> aika
[1] 1 2 3 1 2 3 1 2 3
> laihdutus<-cbind(paino,hlo,aika)
> laihdutus
      paino hlo aika
 [1,] 110.0   1    1
 [2,] 108.0   1    2
 [3,] 107.5   1    3
 [4,]  98.0   2    1
```

```
 [5,]  98.3   2    2
 [6,]  96.0   2    3
 [7,]  74.0   3    1
 [8,]  73.0   3    2
 [9,]  70.0   3    3
```

Nyt lasketaan muutamia esimerkkejä tästä matriisista. Esimerkikiksi komento

```
> apply(laihdutus,1,mean)
[1] 37.33333 37.00000 37.16667 33.66667 34.10000 33.66667 26.00000 26.00000
[9] 25.33333
```

laskee kunkin rivin alkioiden keskiarvon. Komennossa ensimmäinen parametri viittaa siis käytettävään aineistoon, toinen parametri "1"viittaa riveihin ja kolmantena parametrinä annetaan funktio, jota kuhunkin riviin halutaan soveltaa. Vastaavasti asettamalla toiseksi parametriksi numeron 2, voisimme suorittaa keskiarvon laskemisen sarakkeittain.

```
> apply(laihdutus,2,mean)
   paino      hlo     aika
92.75556  2.00000  2.00000
```

tapply-komento on usein erittäin hyödyllinen

```
> tapply(paino,hlo,mean)
        1         2         3
108.50000  97.43333  72.33333
> tapply(paino,aika,sum)
    1     2     3
282.0 279.3 273.5
```

Näin saadaan aineistosta laskettua esimerkiksi kunkin henkilön painon keskiarvo tai henkilöiden yhteispaino kunakin aikana erikseen. Tässä ensimmäinen parametri on muuttuja/lista josta tietoja halutaan laskea, toinen parametri on faktori-tyypin muuttuja (saman pituinen, kuin ensimmäisen parametrin muuttuja) ja kolmas parametri on funktio, jota muuttujiin halutaan soveltaa. Muitten vastaavien komentojen toimintaan, *lapply, sapply, mapply* voi tutustua R:n helppien avulla.

5 Lineaarisista malleista ja grafiikasta

Matriisimerkinnöin lineaarinen regressiomalli voidaan esittää, muodossa

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{0}$$

missä mallimatriisi $\mathbf{X}$ sisältää selittäjät ja $\boldsymbol{\beta}$ sisältää tuntemattomat parametrit. $\boldsymbol{\epsilon}$ on malliin liittyvien satunnaisvirheiden muodostama vektori. Malliin liittyvät oletukset voidaan esittää matriisimuodossa $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma^2\mathbf{I}_n)$, eli satunnaisvirheiden oletetaan noudattavan normaalijakaumaa ja olevan keskenään korreloimattomia.

Yksinkertaista ja selkeää selitystä lineerisista malleista ja niiden laadinnasta tarjoaa Jari Oksasen Internet opas. Tässä ei syvällisemmin mallintamiseen puututa, vaan tyydytään toteamaan, että R on oiva apu myös mallintamisessa. Lineaaristen mallien laadintaa varten on käytössä komento *lm*. Lineaarinen malli jossa y-muuttujaa selitetään x:llä kirjoitetaan R:ssä muotoon $lm(y \sim x)$. Tämä malli sisältää automaattisesti myös yleisvakion. Jos se haluttaisiin välttämättä poistaa pitäisi kirjoittaa $lm(y \sim x - 1)$. Myös monimutkaisempia malleja on helppo muodostaa. Esimerkiksi toisen asteen käyrän sovittaminen tapahtuisi komennolla $lm(y \sim x + I(x^2))$. Käydään läpi yksinkertainen esimerkki *lm* komennon käyttämisestä, jossa muodostetaan esimerkinomaisesti toisen asteen malli. Estimoitu malli on muotoa

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \epsilon_i \tag{1}$$

Käytetään tähän valmista *cars* aineistoa, joka sisältää tiedot autojen nopeuksista ja vastaavista pysähtymismatkoista. Hauskinta on se, että tämä aineisto on koottu jo 1920-luvulla. Aineistossa on kaksi numeerista muuttujaa: nopeus (mailia tunnissa), pysähtymismatka (jalkoja). Havaintoja on kaiken kaikkiaan 50. Älkäämme kiinnittäkö nyt huomiota siihen, että sinänsä toisen asteen polynomin sovittaminen kyseiseen aineistoon ei vaikuta järkevältä, sillä riippuvuus auton nopeuden ja pysähtymismatkan välillä on lineaarista.

```
> data(cars)
> attach(cars)
> names(cars)
```

```
[1] "speed" "dist"
> malli<-lm(dist~speed+I(speed^2))
> malli

Call:
lm(formula = dist ~ speed + I(speed^2))

Coefficients:
(Intercept)        speed   I(speed^2)
    2.47014      0.91329      0.09996
```

Tarkemman tulostuksen mallista saa komennolla *summary*.

```
> summary(malli)

Call:
lm(formula = dist ~ speed + I(speed^2))

Residuals:
    Min      1Q  Median      3Q     Max
-28.720  -9.184  -3.188   4.628  45.152

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.47014   14.81716   0.167    0.868
speed        0.91329    2.03422   0.449    0.656
I(speed^2)   0.09996    0.06597   1.515    0.136

Residual standard error: 15.18 on 47 degrees of freedom
Multiple R-Squared: 0.6673,     Adjusted R-squared: 0.6532
F-statistic: 47.14 on 2 and 47 DF,  p-value: 5.852e-12
```

Nyt esimerkiksi mallin parametrien estimaatit saadaan poimittua komennolla

```
> malli$coefficients
(Intercept)       speed  I(speed^2)
  2.4701378   0.9132876   0.0999593
```

5.1 Grafiikasta

R:ssä on monipuoliset grafiikkaominaisuudet. Sen lisäksi että normaalien pylväsdiagrammien, histogrammien ja pisteparvien esittäminen on suhteellisen helppoa, on mahdollsita piirtää myös monimutkaisempia kuvia, kuten aikasarjoja ja kolmiulotteisia kuvia. R:n tarjoamiin grafiikkamahdollisuuksiin kannattaa tutustua katsomalla demoja. Muun muassa seuraavat antavat hyvän kuvan R:n monipuolisuudesta piirto-ohjelmana.

```
> demo(graphics)
> demo(image)
> demo(persp)
```

Piirrettään edellä muodostetusta toisen asteen mallista kuva R:n *plot*-komennolla. *par*-komennon avulla saadaan useampi kuva samalle sivulle, tässä tapauksessa 2×2 eli 4.

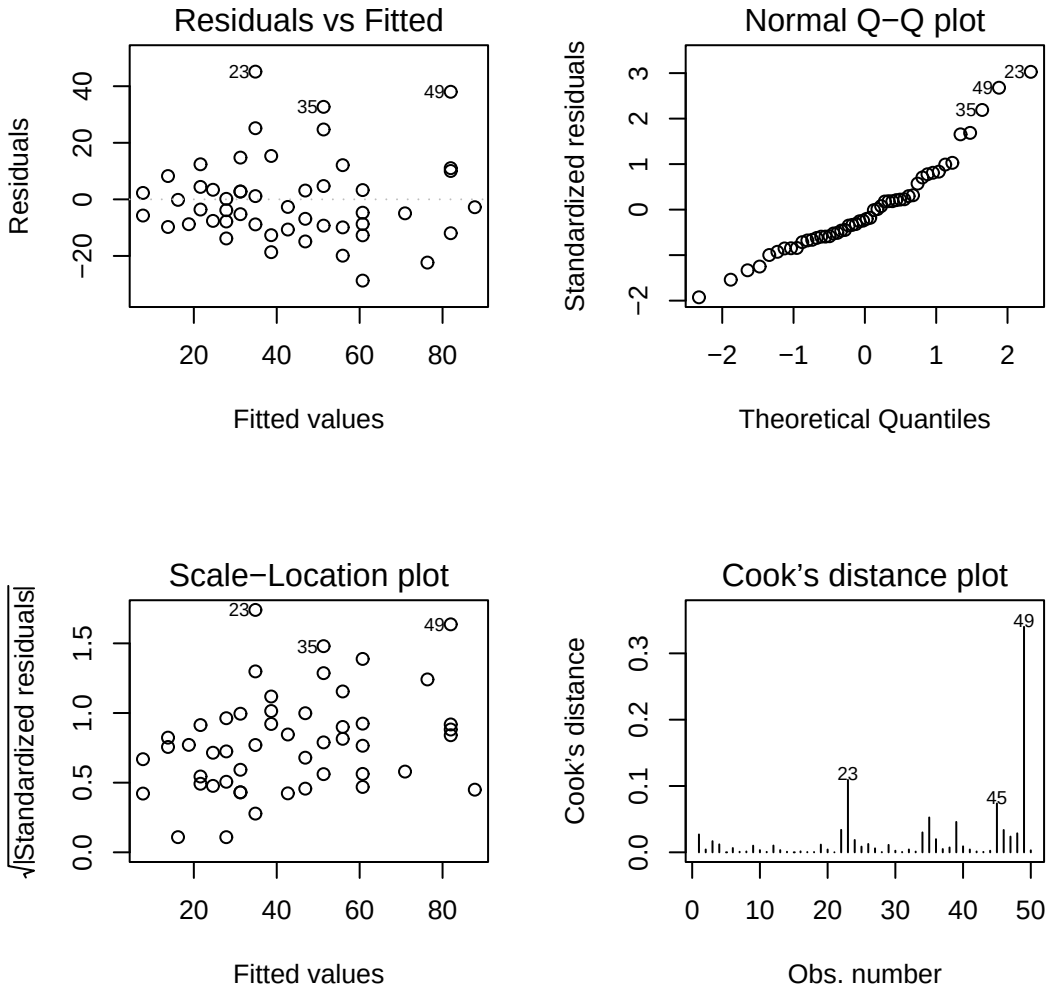
```
> par(mfrow=c(2,2))
> plot(malli)
```

Muutama sananen kuvista lienee paikallaan. Nämä kuvaukset pohjautuvat Jari Oksasen oppaaseen, joskin mallin sopivuudesta aineistoon ei tässä kiinnitetä huomiota.

1. Residuals vs Fitted: Regression residuaalit vastaan ennustetut arvot. Mikäli malli sopii, residuaalit muodostavat tasaisen vyön suoran Residuals = 0 ympärille. Meillä näkyy olevan muutama hyvin poikkeava arvo; havainnot 23, 35 ja 49 on merkitty erityisen poikkeaviksi.

2. Normal Q-Q plot: Vaaka-akselilla on normaalijakauman mukaiset teoreettiset kvantiilit, pystyakselilla standardisoidut residuaalit. Mikäli residuaalit ovat normaalisesti jakautuneita, kuvassa on suora viiva. Mitä ilmeisimmin normaalisuusoletusta vastaan on rikottu. Etenkin havainnot 23, 35 ja 49 käyttäytyvät huonosti.

3. Scale-Location plot: Kuten osakuva 1, mutta nyt näytetään vain poikkeaman suuruus, ei suuntaa. Etenkin havainnot 23, 35 ja 49 käyttäytyvät huonosti.



Kuva 1: R:n oletusarvoisesti mallista piirtämä kuva.

4. Cook's distance plot: Cookin etäisyys mittaa sitä, kuinka vaikuttava kukin havainto on. Suuri Cookin etäisyys tarkoittaa, että yksittäisellä pisteellä on suuri vaikutus regressiosuoraan. Etenkin pisteet 23, 45 ja 49 näyttävät vaikuttavan tähän suoraan suuresti.

5.2 Sekamalleista

Yleisessä muodossa sekamalli voidaan kirjoittaa

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \boldsymbol{\epsilon}, \tag{2}$$

missä $\mathbf{Z}$ on satunnaisvaikutusten suunnittelumatriisi ja $\mathbf{u}$ on satunnaisvaikutusten vektori. Sekamallissa määritellään, että $E(\mathbf{y}) = \mathbf{X}\boldsymbol{\beta}$, $E(\mathbf{u}) = \mathbf{0}$ ja $E(\boldsymbol{\epsilon}) = \mathbf{0}$. Lisäksi oletetaan, että satunnaisvirheet ja satunnaisvaikutukset ovat riippumattomia, jolloin $Cov(\mathbf{u}, \boldsymbol{\epsilon}) = \mathbf{0}$ Sekamallien mallintamiseen tarkoitettu funktio *lme* löytyy *nlme*-kirjastosta, joka on siis ladattava käyttöön *library(nlme)*-komennolla. Käydään sekamallin muodostamista läpi esimerkin avulla. Ladataan ensin käyttöön pakkauksesta nlme *Orthodont* aineisto.

```
> library(nlme)
> data(Orthodont)
> attach(Orthodont)
> names(Orthodont)
[1] "distance" "age"      "Subject"  "Sex"
```

Aineistossa yhdeltätoista tytöltä ja kuudeltatoista pojalta on mitattu tietty hampaistoon liittyvä etäisyys 8-, 10-, 12-, ja 14-vuoden iässä. Etäisyyden oletetaan kasvavan lineaarisesti ajan funktiona.

```
> levels(Sex)
[1] "Male"   "Female"
```

Poimitaan naispuoliset henkilöt omaksi ainoistokseen.

```
> Orthofem<-Orthodont[Sex=="Female",]
```

Muodostetaan erittäin yksinkertainen sekamalli, jossa etäisyyttä selitetään iällä. Nyt mallilausekkeessa annetaan vain kiinteät vaikutukset. Satunnaiset tekijät annetaan erikseen määritteellä *random*. Nyt random-osa kertoo, että keskiarvo (1) on ehdollinen (|) satunnaistekijälle Subject (eli yksilö).

```
> femlin<-lme(distance~age,data=Orthofem,random=~1|Subject)
> summary(femlin)
Linear mixed-effects model fit by REML
```

```
 Data: Orthofem
       AIC     BIC    logLik
  149.2183 156.169 -70.60916

Random effects:
 Formula: ~1 | Subject
        (Intercept)  Residual
StdDev:     2.06847 0.7800331

Fixed effects: distance ~ age
                Value Std.Error DF   t-value p-value
(Intercept) 17.372727 0.8587419 32 20.230440       0
age          0.479545 0.0525898 32  9.118598       0
 Correlation:
    (Intr)
age -0.674

Standardized Within-Group Residuals:
       Min          Q1         Med          Q3         Max
-2.2736479 -0.7090164  0.1728237  0.4122128  1.6325181

Number of Observations: 44
Number of Groups: 11
```

Nyt mallista voidaan poimia esimerkiksi kiinteiden ja satunnaisvaikutusten estimaattiarvot

```
> femlin$coefficients
$fixed
(Intercept)         age
 17.3727273   0.4795455

$random
$random$Subject
    (Intercept)
F10 -4.00532867
```

```
F09 -1.47044943
F06 -1.47044943
F01 -1.22903236
F05 -0.02194701
F07  0.34017860
F02  0.34017860
F08  0.70230420
F03  1.06442981
F04  2.15080663
F11  3.59930905
```

On syytä huomioida, että *summary*-komento antaa estimaattien keskihajonnat, eikä suinkaan variansseja. Mallin parametreja määrittelemällä estimointi voidaan suorittaa esimerkiksi suurimman uskottavuuden (ML) tai rajoitetun suurimman uskottavuuden menetelmällä (REML), jota R oletusarvoisesti käyttää. Mallin määreenä voidaan myös satunnaisvaikutusten kovarianssimatriisille $Var(\mathbf{u})$ antaa erilaisia rakenteita, kuten tasakorrelaatio, indentiteetti, diagonaali, blokki-diagonaali tai yleinen rekenne. Vastaavat R-komennot ovat *pdCompsym*, *pdIdent*, *pdDiag*, *pdBlocked*, *pdSymm*. Kirjasto *nlme* tarjoaa erilaisia rakenteita myös satunnaisvirheiden kovarianssimatriisille $Var(\boldsymbol{\epsilon})$. Esimerkiksi voidaan käyttää seuraavia rakenteita: tasakorrelaatio, ar(1)-rakenne, yleinen, ARMA-prosessista syntyvä rakenne ja lineaarinen rakenne. R:llä nämä ilmaistaan komennoin *corCompSymm*, *corAR1*, *corSymm*, *corARMA* ja *corLin*. Muodostetaan esimerkiksi malli, jossa satunnaisvaikutusten kovarianssimatriisille annetaan diagonaalirakenne ja estimoinnissa käytetään suurimman uskottavuuden menetelmää.

```
> femDiag<-lme(distance~age,data=Orthofem,random=pdDiag(~age),method="ML")
> summary(femDiag)
Linear mixed-effects model fit by maximum likelihood
 Data: Orthofem
       AIC      BIC    logLik
  144.6821 153.6031 -67.34106

Random effects:
 Formula: ~age | Subject
```

```
 Structure: Diagonal
        (Intercept)      age  Residual
StdDev:    1.482869 0.124882 0.6884841

Fixed effects: distance ~ age
                Value Std.Error DF   t-value p-value
(Intercept) 17.372727 0.7027271 32 24.721870       0
age          0.479545 0.0611758 32  7.838809       0
 Correlation:
    (Intr)
age -0.578

Standardized Within-Group Residuals:
        Min           Q1          Med           Q3          Max
-2.08314737 -0.50739001  0.08990427  0.52914524  1.56922904

Number of Observations: 44
Number of Groups: 11
```

Nyt estimoitu satunnaisvaikutusten kovarianssimatriisi, $Var(\mathbf{u})$, on

$$\begin{pmatrix} 1.482869^2 & 0 \\ 0 & 0.124882^2 \end{pmatrix} = \begin{pmatrix} 2.198900 & 0 \\ 0 & 0.01559551 \end{pmatrix} \tag{3}$$

Mallissa $Var(\boldsymbol{\epsilon}) = \sigma^2\mathbf{I} = 0.6884841^2\mathbf{I} = 0.4740104\mathbf{I}$ Muodostetaan seuraavaksi malleja, joissa satunnaisvirheiden kovarianssimatriisia on mallinnettu. Kovarianssimatriisia mallintaville funktioille annetaan kaksi argumenttia arvo ja muoto. Argumenttia *fixed* voidaan käyttää, jos halutaan kiinteä kovarianssimatriisi. Tällöin määritellään *FIXED=TRUE*, koska oletusarvoisesti *FIXED=FALSE*. Jos esimerkiksi halutaan käyttää tasakorrelaatiorakennetta Orthodont aineistolle voidaan määritellä

```
> cs1<-corCompSymm(value=0.3,form=~1|Subject)
> cs1<-Initialize(cs1,data=Orthodont)
> cs1
Correlation structure of class corCompSymm representing
Rho
0.3
```

Esimerkiksi ensimmäiselle miehelle matriisi näyttää seuraavalta

```
> corMatrix(cs1)
$M01
     [,1] [,2] [,3] [,4]
[1,]  1.0  0.3  0.3  0.3
[2,]  0.3  1.0  0.3  0.3
[3,]  0.3  0.3  1.0  0.3
[4,]  0.3  0.3  0.3  1.0
```

Initialize-komento ei ole välttämätön varsinaisen mallin laadinnan kannalta, mutta sitä tarvitaan saadaksemme näkyviin esimerkiksi juuri kovarianssirakenteen muodon ensimmäiselle miehelle. Muodostetaan nyt varsinainen malli. Huomaa, että *random* osaa ei nyt ole erikseen määritelty, jolloin R käyttää oletusarvona kiinteän eli *fixed*-osan määrittelyä.

```
> OrthTas<-lme(distance~age,data=Orthodont,correlation=cs1)
> summary(OrthTas)
Linear mixed-effects model fit by REML
 Data: Orthodont
       AIC      BIC    logLik
  456.6367 475.2808 -221.3183

Random effects:
 Formula: ~age | Subject
 Structure: General positive-definite
            StdDev    Corr
(Intercept) 2.1370661 (Intr)
age         0.2264340 -0.663
Residual    1.6013036

Correlation Structure: Compound symmetry
 Formula: ~1 | Subject
 Parameter estimate(s):
      Rho
0.3307065
Fixed effects: distance ~ age
```

```
                Value Std.Error DF  t-value p-value
(Intercept) 16.761111 0.7752419 80 21.62049       0
age          0.660185 0.0712537 80  9.26527       0
 Correlation:
    (Intr)
age -0.848

Standardized Within-Group Residuals:
        Min           Q1          Med           Q3          Max
-2.87625539 -0.39829180 -0.01722205  0.42874852  3.26698618

Number of Observations: 108
Number of Groups: 27
```

Tulosteesta näemme satunnaisvaikutusten keskihajonnat ja korrelaation, joten satunnaisvaikutusten kovarianssimatriisi on muotoa

$$\begin{pmatrix} 4.567052 & -0.3208286 \\ -0.3208286 & 0.05127236 \end{pmatrix} \tag{4}$$

Muodostetaan esimerkin vuoksi vielä yleinen rakenne.

```
> sym1<-corSymm(value=c(0.2,0.1,-0.1,0,0.2,0),form=~1|Subject)
> sym1<-Initialize(sym1,data=Orthodont)
> sym1
Correlation structure of class corSymm representing
 Correlation:
  1    2    3
2  0.2
3  0.1  0.0
4 -0.1  0.2  0.0
> OrthSym<-lme(distance~age,data=Orthodont,correlation=sym1)
> summary(OrthSym)
Linear mixed-effects model fit by REML
 Data: Orthodont
       AIC      BIC    logLik
  461.0076 492.9689 -218.5038
```

```
Random effects:
 Formula: ~age | Subject
 Structure: General positive-definite
            StdDev    Corr
(Intercept) 2.8957249 (Intr)
age         0.2742948 -0.747
Residual    1.3181699

Correlation Structure: General
 Formula: ~1 | Subject
 Parameter estimate(s):
 Correlation:
  1      2      3
2 -0.133
3  0.245 -0.215
4  0.112  0.419  0.148
Fixed effects: distance ~ age
                Value Std.Error DF   t-value p-value
(Intercept) 16.577302 0.7506068 80 22.085200       0
age          0.674176 0.0705013 80  9.562609       0
 Correlation:
    (Intr)
age -0.842

Standardized Within-Group Residuals:
        Min          Q1         Med          Q3         Max
-2.88933500 -0.45845446  0.02377598  0.42452588  3.70656815

Number of Observations: 108
Number of Groups: 27
```

6 Saatavilla olevaa R-materiaalia

6.1 Jari Oksasen opas ekologeille

Tässä osiossa keskitymme Internetistä löytyvään R-materiaalin. Ymmärrettävästi suurin osa tästä aineistosta on englanninkielisiä. Hyviä suomalaisia opassivuja voi olla vaikeakin löytää. Eräs suhteellisen kattava sivusto on Oulun yliopiston Biologian laitoksen professori Jari Oksasen tekemä opas ekologeille. Tämä opas löytyy osoitteesta http://cc.oulu.fi/∼jarioksa/opetus/rekola/). Oksanen kuvaa internet sivuillaan opasta seuraavasti. "R: opas ekologeille"on tarkoitettu oheislukemistoksi Rannan, Ridan ja Koukin Biometrialle. Oppaassa käsitellään Biometrian kirjan (mielestäni) keskeisimmät menetelmät, joiden esimerkit analysoidaan R-ohjelmalla. (Oksanen 2003.) Kuten Oksanen itsekin toteaa hän on valinnut esiteltäväksi laajasta R-ohjelmasta vain pienen osan, joka sisältää keskeisimmät ekologien tarvitsemat menetelmät. Näin tämä opas ei välttämättä tarjoaa kaikkea oleellista tietoa muiden aineiden ja alojen opiskelijoille, mutta monta hyödyllistä seikkaa kuitenkin. Erityisen positiivista on asioiden yksinkertainen selittäminen ja yksityiskohtaiset ohjeet R:n käytön aloittamiseksi.

6.2 R-ohjelmiston kotisivujen sisällöstä

R-ohjelmiston kotisivut sijaitsevat osoitteessa http://www.r-project.org/index.html. Näiltä sivuilta voi kuka tahansa ladata R-ohjelman käyttöönsä. Täältä löytyy taustatietoa R:stä, sen kehityksestä ja kehittäjistä ja uusimpia uutisia. Näillä sivuilla kuka tahansa voi esimerkiksi tehdä ilmoituksen R:stä löytämistään virheistä ja näin auttaa kehittämään R:ää kohti täydellisempää ohjelmistoa. Sivuilta löytyy myös muutamia R-oppaita. Ensinnäkin johdanto-osa, joka esittelee R-kieltä ja miten käyttää R:ää tilastollisiin analyyseihin ja grafiikan tekemiseen. Toisessa oppaassa paneudutaan syvällisemmin omien funktioiden ohjelmoimiseen R:n avulla. Kolmas opas neuvoo kuinka luoda omia pakkauksia. Neljäs opas käsittelee materiaalin liikuttamista R:ään ja R:stä pois. Tämä on hyödyllinen taito, sillä usein esimerkiksi valmiiksi saatu aineisto on jollain toisella ohjelmalla tehty. Viides opas käsittelee R:n asentamista ja hallintaa.

6.3 Muuta materiaalia

Tehdään lopuksi pintapuolinen silmäys internetistä löytyvään suomenkieliseen R-materiaaliin. Jyväskylän yliopistossa näytetään olevan aktiivisia R-materiaalin suhteen, koska sitä kautta löytyy kaksi hyvää R-opasta. Seuraavassa siis muutamia linkkejä, joista osa tosin sisältää vain muutaman komennon tai yleistä kuvausta R-ohjelmistosta. Kattavammatkin oppaat ovat yleensä vain tiettyyn alaan ja sen tarpeisiin keskittyneitä.

1. *http://www.jyu.fi/finance/ohjelmistot/R.pdf*

Tästä osoitteesta löytyy professori Antti Penttisen kurssimoniste S-plus/R-ohjelmointi, 2002. Moniste on jaettu kahdeksaan lukuun, joista kaksi ensimmäistä keskittyy esittelyesimerkkeihin sekä vektori- ja matriisikomentoihin. Luvussa 2.4 esitellään myös datakehikon (*data.frame*) muodostaminen ja luvussa 3 käsitellään lisää datan lukemista ja tulostamista. Luku 4 keskittyy kokonaisuudessaan R:n monipuoliseen grafiikkaan vaikkakin pintapuolisesti. Luvussa 5 Penttinen käsittelee jakaumia ja luvussa 6 omien funktioiden kirjoittamista, joihin tässä työssä ei ole puututtu lainkaan. Luvussa seitsemän käsitellään matriisialgebran soveltamista lineaarisen mallin laskentaan. Viimeisessä luvussa on joitakin hyödylliseksi mainittuja lisäpiirteitä R:stä, muun muassa keinoja etsiä virheitä virhetilanteissa.

2. *http://www.jyu.fi/finance/ohjelmistot/RFinance.pdf*

Toinen Jyväskylän yliopiston kautta löytyvä R-opas on Risto Tammelan kirjoittama. Tämä opas on taloustieteilijöille suunnattu, mutta myös muiden alojen ihmisille voi hyödyllisiä komentoja ja ohjeita tästä julkaisusta löytyä. Esimerkiksi käyttökelpoisia ovat ohjeet datan siirtämisestä Excelin ja R:n välillä. Tammela itse kuvaa opastaan seuraavasti. ”Tämä opus ja sen funktiot on tarkoitettu vain esimerkeiksi ja innostuksen antajiksi. Tarkoituksena oli osoittaa, että tämä tilastotieteilijöiden työkalu R taipuu rahoituksen ongelmiin.” (Tammela 2003.)

3. *http://www.helsinki.fi/∼muurimak/R/*

Tästä osoitteesta löytyy Helsingin yliopiston Perttu Muurimäen kirjoittamaa R-tekstiä. Sivusto sisältää lähinnä sekalaisia huomioita R:stä, mutta on varmasti selaamisen arvoinen. Lisäksi sivustolta löytyy muutamia linkkejä muuhun R-materiaaliin.

4. *http://noppa5.pc.helsinki.fi/S/*

Tällä sivulla on Juha Purasen, niinikään Helsingin yliopistosta, datoja ja R-koodia. Koodi ei sisällä sen tarkempia selityksiä, joten siitä ei liene apua ainakaan R:n käyttöä aloittelevalle. Omia funktioita kirjoittaessa voi kuitenkin tältä sivulta löytyä hyödyllisiä vinkkejä.

5. *http://www.helsinki.fi/atk/lehdet/401/Ilmainen_R-ohjelmisto.html*
http://www.helsinki.fi/atk/lehdet/401/Atk-osasto_testasi.html

Ensimmäisen linkin takaa löytyy Antti Nevanlinnan kirjoittama lyhyt artikkeli R:stä. Sivulla on muun muassa lyhyt käyttöesimerkki R-istunnosta Windows NT:llä. Muutamia uusia komentoja voi täältäkin löytyä, mutta ei mitään radikaalisti erilaista ja uutta. Toisen linkin takaa löytyy mielenkiintoinen tilastoohjelmien vertailuun, jossa R:kin on mukana.

6. *http://www.csc.fi/lehdet/atcsc/atcsc3-2001/r.pdf*

Jarno Tuimala on kirjoittanut lyhyehkön artikkelin R-ohjelmistosta ja mikrosiruanalyysistä. Tästäkin artikkelista saattaa lukija löytää joitakin yksittäisiä hyödyllisiä komentoja.

7. *http://www.csc.fi/lehdet/atcsc/atcsc1-2002/r.pdf*

Tämä on niinikään Jarno Tuimalan kirjoittama artikkeli. Parisivuinen teksti käsittelee R-ohjelmistoa ja monimuuttujamenetelmiä.