R:
Opas ekologeille

Jari Oksanen

Biologian laitos
Oulun yliopisto

Versio 1.0
21. tammikuuta 2003

Saatteeksi

Tämä opas johdattaa lukijan — toivottavasti lempeästi — R-ohjelmaan. Opas on syntynyt välittömään opetustarpeeseen kun olen joutunut opetttamaan biometrian perusteita ja tilasto-ohjelmia ekologisesti suuntautuneille biologian opiskelijoille, lähtökohtana suomalainen biometrian oppikirja (Ranta *et al.*, 1989). Oppaan esimerkit ovat peräisin Biometrian oppikirjasta, mutta tällä kertaa ne analysoidaan R-ohjelmalla. Lienee hyödyllisintä lukea tätä opasta rinnan Biometrian kirjan kanssa, etenkin kun useimmat oppaan analyyseistä poikkeavat alkuperäisestä.

R on todella laaja ja voimakas ohjelma. Tämä opas kattaa vain pienen osan R:stä. Olen valinnut esiteltäväksi vain mielestäni keskeisimmät ekologien tarvitsemat menetelmät. Käytännössä painopiste on R:n lineaarisissa malleissa, jotka ovat ekologisten koe- ja havaintoaineistojen käsittelyn tärkein työkalu. Olen pyrkinyt kirjoittamaan oppaan melko käytännönläheiseksi. R:n ohjelmointia käsitellään vain hyvin pintapuolisesti. Pääpaino on komennoissa, joilla aineistosta saa mahdollisimman nopeasti perusanalyysit. Alkututustumisen jälkeen käyttäjä siirtyy toivottavasti mahdollisimman pian perusteellisempien oppaiden ja syvempien analyysien pariin.

Korjaukset ja kommentit ovat tervetulleita osoitteeseen

`jari.oksanen@oulu.fi`

Tämän oppaan aineistot ovat saatavissa R-paketissa `rekola`. Tämä paketti samoin kuin oppaan uusin versio ja mahdolliset lisätiedot ovat saatavilla verkko-osoitteestani

`http://cc.oulu.fi/~jarioksa/`

Esa Läärän lukuisat kommentit ovat oleellisesti parantaneet tätä opasta.

Jari Oksanen
Oulu, 21. tammikuuta 2003

Sisältö

1 Mikä on R?

1.1 Tilastollinen ohjelmointiympäristö

R ei ole vain tilastollinen ohjelma vaan on tilastollinen *ohjelmointiympäristö*. Se ei koostu vain erillisistä ohjelmista, vaan muodostaa eheän kokonaisuuden, jossa käyttäjä voi myös helposti ohjelmoida omia tilastollisia menetelmiään. Vaikka itsellä ei olisikaan haluja menetelmien ohjelmointiin, voi luottaa siihen, että joku muu on ohjelmoinnut ja julkistanut uudetkin temput. Tämän ansiosta R on hyvin nopeasti kehittyvä ja nykyaikainen ohjelmisto. Lisäksi R:ssä on hyvät graafiset ominaisuudet.

R muistuttaa suuresti S-kieltä ja sen pohjalta kehitettyä S-PLUS-ohjelmaa. S on alkujaan AT&T laboratoriossa kehitetty tilastollinen ohjelmointikieli, joka on yksinoikeudella lisensoitu S-PLUS -ohjelman tuottajalle. R on sisäiseltä toteutukseltaan hyvin erilainen kuin S, mutta käyttäjän kannalta kielet ovat samanlaiset. Niinpä S- ja S-PLUS -ohjelmien oppaat soveltuvat melko suoraan myös R:n käyttöön (esim. Venables & Ripley, 1999)

R on komentopohjainen ohjelma. Noviisista tämä usein tuntuu vastenmieliseltä ja hankalalta verrattuna näennäisesti helppokäyttöisiin hiirellä käytettäviin ohjelmiin. Kieltämättä komentojen opiskelusta on oma vaivansa, mutta itse asiassa perusanalyysit voi tehdä melko helposti. Suuri osa tämän monisteen analyyseistä tehdään yhdellä komennolla (`lm`). Tilastollisen ohjelman käytössä pääongelma on tilastollinen ymmärrys — hiirivetoisella ohjelmalla voi helposti tehdä analyysejä, joita ei itsekään ymmärrä (tosin se on mahdollista myös R:llä, mutta vaatii hieman enemmän vaivaa ylittää omat tietonsa). Ekologi joutuu käyttämään etupäässä regressio- ja varianssianalyysejä, ja niissä pääongelma on mallin (eli muuttujien suhteiden) määrittely. Tämä on usein paljon helpompi tehdä kirjoittamalla yksinkertainen mallilauseke kuin valitsemalla vaihtoehtoja hiirellä.

1.2 Ohjelman hankkiminen ja asennus

R on vapaa ohjelma: kuka tahansa saa vapaasti ja ilmaiseksi hankkia ohjelman, asentaa sen kuinka moneen tietokoneeseen tahansa, antaa kopioita kelle tahansa ja käyttää ohjelmaa mihin tahansa tarkoitukseen, omalla vastuullaan. Tällä hetkellä ohjelman kotisivut ovat osoitteessa `http://cran.r-project.org/`. R on saatavilla sekä ohjelman lähdekoodina että valmiiksi käännettynä monille käyttöjärjestelmille. Tämä opas on kirjoitettu LINUX-järjestelmän versiolle, mutta R on saatavilla myös Windowsiin ja Macintoshiin. Näiden versioiden käyttäminen on hyvin samanlaista, vaikka kullakin alustalla on omat erityispiirteensä.

R:n kotisivuilla on ajantasaiset ohjeet ohjelman asentamiseksi eri käyttöjärjestelmiin. Yleensä on helpointa hankkia valmiiksi käännetty binaaritiedosto. R on suunniteltu siten, että siinä on melko pienikokoinen yhteinen perusosa (`base`), minkä lisäksi kannattaa yleensä hankkia joitain laajennusosia (`packages`). Esimerkiksi tässä oppaassa käsiteltävät aineistot ovat saatavissa paketissa `rekola`. R kannattaa hankkia nimenomaan omilta kotisivuiltaan, joissa on aina tarjolla tuorein ja paras versio. Esimerkiksi monessa LINUX-jakelupaketissa on mukana R, mutta yleensä väljähtäneenä versiona. R-ohjelmassa on mainiot valmiudet laajennuspakettien hallintaan ja jopa automaattiseen ajantasaistamiseen.

2 R–istunto

Oppaan esimerkeissä R:n kehote kirjoitetaan "`>`". Jos komento jatkuu toiselle riville, R:n jatkokehote kirjoitetaan "`+`". Käyttöjärjestelmässä annettujen komentojen kehote on "`$`". Joissain tapauksissa R-komentoihin on lisätty *kommentti*, joka seuraa merkkiä "`#`". Yksittäisen näppäimen painallukset on merkitty kehykseen, esimerkiksi r-näppäimen painaminen merkitään R. Jos useampaa näppäintä on painettava yhtaikaa, on ne yhdistetty väliviivalla: CTRL–ALT–DEL.

2.1 Istunnon aloitus ja lopetus

R-istunto aloitetaan ikonia klikkaamalla tai LINUXissa[1] kirjoittamalla

```
$ R
```

Jos istunto on aloitettu ikonista, voi sen lopettaa ikkunavalikosta, mutta komentotulkissa avattu istunto on myös suljettava komennolla:

```
> q()
```

Tämäkin lyhyt esimerkki paljastaa joitain oleellisia (ja kenties kiusallisia) piirteitä R:stä:

- R on tarkka merkkilajista: komento on kirjoitettava pienellä kirjaimella. Komento `Q()` ei toimisi tai jos toimisi, tekisi jotain muuta kuin lopettaisi istunnon.
- Komennon tunnistaa siitä, että sitä seuraavat sulut "`()`". Pelkkä `q` tulostaisi komennon ohjelmakoodin näytölle. Tällä kertaa komento annettiin ilman sulkujen sisään kirjoitettavia määritteitä, mutta sulut oli silti kirjoitettava näkyviin (tosin `q()` kysyy puuttuvien määritteiden arvot mikäli niitä ei ole annettu).

2.2 Alustavat toimenpiteet

Lopetettaessa istunto, R kysyi, talletetaanko istunto nykyiseen työhakemistoon. Mikäli vastaus on myönteinen, R kirjoittaa työhakemistoon tiedostot `.RData` ja `.Rhistory` (nämä ovat ainakin UNIX-nimet: Windows saattaa käyttää toisenlaisia nimiä). UNIX-järjestelmässä pisteellä alkava tiedosto on piilotiedosto, joka ei näy normaalissa hakemistolistauksessa, ellei sitä pyydetä määritteellä "`ls -a`". Tämän takia jokainen "projekti" kannattaa aloittaa omassa hakemistossaan, jolloin talletetut istunnot ja komennot säilyvät siististi omissa lokeroissaan. Projektin tarvitsemat datatiedostot kannattaa myös siirtää aluksi tähän työhakemistoon.

Projekti kannattaa aloittaa luomalla uusi työhakemisto ja siirtymällä sinne:

```
$ mkdir kurssi
$ cd kurssi
$ R
> q()
$
```

Macintoshissa ja Windowsissa periaate on samanlainen, mutta hakemistojen hallinta tapahtuu todennäköisesti graafisesti.

2.3 Apua!

Graafisessa R–ikkunassa ohjeita saa **Help**–valikosta, mutta R-ohjelman funktioista voi pyytää ohjeita myös komennoilla

[1] Myös Windowsissa istunnon voi aloittaa komentotulkista, mutta komento on `Rterm`

```
> help(komento)
> ?komento
```

Ohjesivut ovat yleensä melko selkeät ja lisäksi niiden lopussa on esimerkkejä, jotka tekemällä saa kuvan komennon mahdollisuuksista. En käsittele R-komentoja erityisen systemaattisesti vaan esittelen vain kuhunkin tehtävään tarvittavia pääominaisuuksia. Kannattaa siis yleensä katsoa uuden komennon ohjesivut, jotta näkee mitä kaikkea komennolla todella voi tehdä.

Tavalliset ohjesivut tulostuvat ruudullinen kerrallaan. Näppäilemällä Q pääsee kesken ohjesivun palaamaan R-komentoihin.

Saadakseen ohjeita komennon käytöstä on tietysti muistettava komento — ja juuri se onkin tuntematon. Mikäli komennosta muistaa tai arvaa osan, voi ohjesivuja etsiä komennoilla `apropos` tai `help.search`. Esimerkiksi kaikki kuvanpiirtokomennon `plot` tyypit löytyvät kirjoittamalla

```
> apropos("plot")
> help.search("plot")
```

Kysytty komento tai sen osa kannattaa todellakin kirjoittaa lainausmerkkeihin, koska näin saattaa löytyä enemmän komentoja. Kiusallista kyllä, lainausmerkkien pois jättäminen voi tuottaa joitain tuloksia, muttei kaikkia.

On myös mahdollista käyttää `help`-järjestelmää graafisen selaimen, LINUXissa yleensä Netscapen kautta. Kun istunnossa annetaan komento

```
> help.start()
```

käynnistyy selain ja näyttää kysytyn ohjetiedoston[2]. Tällä tavoin varsinainen istuntoikkuna säästyy työkäyttöön. Lisäksi HTML-tyyppisessä `help`-järjestelmässä voi navigoida ristiviittausten avulla sekä käyttää hakutoimintoja, joten tuntemattomankin komennon löytäminen helpottuu. Kun graafinen ohjeikkuna on avattu, kaikki R:ssä annetut ohjekyselyt näkyvät vain selaimessa.

2.4 Komennot

R-komentoa seuraa aina sulut `()`, joiden sisällä annetaan komennon lisämääreet. Kaikki mahdolliset lisämääreet näkee `help`-komennon avulla. Periaatteessa määritteet voi antaa luettelemalla samassa järjestyksessä kuin `help` ne luetteli. R:ssä on onneksi kaksi käyttöä helpottavaa piirrettä:

- Nimetyt parametrit, joten arvot voi ilmoittaa missä järjestyksessä tahansa kirjoittamalla `(..., parametri=arvo, ...)`.
- Oletusarvot, joten vain ne parametrit tarvitsee mainita, joiden arvoja halutaan muuttaa.

Esimerkiksi käyköön `q`-komennon ohjesivut:

```
Terminate an R Session

Description:

     The function 'quit' or its alias 'q' terminate the current R
     session.

Usage:

     quit(save = "default", status = 0, runLast = TRUE)
        q(save = "default", status = 0, runLast = TRUE)

     ... loput poistettu ...
```

[2] Ei välttämättä Windowsissa.

Ohjesivusta käy ilmi, että komento `q` on itse asiassa synonyymi pidemmälle komennolle `quit`, jota myös voisi käyttää istunnon lopettamiseen. Komennolla voi näemmä olla kolme parametria, joista ensimmäinen "`save`" määrää, talletetaanko istunto. Sen oletusvastauksen ("`default`") on määritellyt R:n ylläpitäjä (yleensä "`ask`" eli kysytään käyttäjältä, talletetaanko tiedosto). Jos käyttäjä haluaa tallettaa istunnon vastaamatta kysymyksiin, hän voi lopettaa istunnon kahdella vaihtoehtoisella tavalla:

```
> q("yes")
> q(save="yes")
```

2.5 Istunnon hallinta

Lopetettaessa R-istunto, ohjelma kysyy talletetaanko istunto. Mikäli tähän vastaa myönteisesti

```
> q()
Save workspace image? [y/n/c]: y
```

kaikki istunnossa annetut komennot tallettuvat tiedostoon `.Rhistory` ja kaikki muuttujat ja aineistot tallettuvat tiedostoon `.RData`. Käynnistettäessä istunto seuraavan kerran, sekä aineisto että entiset komennot luetaan muistiin. Komento `ls()` (tai vaihtoehtoisesti `objects()`) antaa luettelon käytettävissä olevista aineistoista ja muuttujista. Vanhoja komentoja voi selata (ainakin LINUX-R:ssä) nuolinäppäimillä $\boxed{\uparrow}$ ja $\boxed{\downarrow}$. Aiemmin annettuja komentoja voi etsiä näppäilemällä $\boxed{\text{CTRL}}$–$\boxed{\text{R}}$ ja kirjoittamalla osan annetusta komennosta. Kun haluttu komento on löytynyt, sen saa käyttöön näppäilemällä $\boxed{\text{ESC}}$, minkä jälkeen komentoa pystyy myös muokkaamaan.

Jos istuntoon alkaa kertyä liikaa epämääräisiä, alkujaan väliaikaisiksi tarkoitettuja muuttujia, ylimääräiset voi poistaa komennolla `rm`. Täysin puhtaaksi työtiedostonsa saa hävittämällä tiedostot `.RData` ja `.Rhistory`. Tähän ei ole juurikaan tarvetta, mikäli kukin osaprojekti on alkujaankin aloitettu omassa alihakemistossaan, sillä nämä tiedostot eivät näy muihin hakemistoihin.

2.6 Lisäpaketit eli kirjastot

Perus-R:ssä on vain kaikkein keskeisimmät menetelmät ja funktiot. Suuri osa R:n arvokkaista ohjelmista on itse asiassa 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=pkg)  # Kirjaston pkg ohjelmat
> library(pkg)       # Ottaa käyttöön kirjaston pkg
> detach(pkg)        # Poistaa käytöstä kirjaston pkg
```

Windowsissa kirjastojen käyttöön otto ja selailu on mahdollista suoraan ikkunavalikosta.

Tämän oppaan esimerkkien ja tehtävien tiedostot ovat paketissa `rekola`, joka on saatavissa kotisivuiltani `http://cc.oulu.fi/~jarioksa/`. Muita oppaassa käytettyjä paketteja ovat yleishyödyllinen `MASS` (Venables & Ripley, 1999) ja monimutkaisissa lineaarisissa malleissa (§7.7) käytetty `nlme`. Pakettia `foreign` tarvitaan lukemaan muilla tilasto- ja taulukkolaskentaohjelmilla tallennettu ulkoinen havaintotiedoston R:ssä käsittelykelpoiseksi datakehikoksi (ks. s. 11).

2.7 Analyysien tallennus ja tulostus

Nykyisellään paljaassa R:ssä ei ole kovin kehittynyttä tapaa tallentaa analyysien tuloksia tiedostoon, mutta esimerkiksi Windowsin Rgui-ikkunan istunto voidaan tallettaa ja tulostaa. Muiden,

esimerkiksi LINUX-käyttäjien kannattaa asentaa (X)Emacs-ohjelma ja ESS (*"Emacs Speaks Statistics"*), jonka avulla emacsista käsin pystyy käyttämään R:ää ja monia muita tilastollisia ohjelmia (S-PLUS, SAS, SPSS...). Kevyempi vaihtoehto on keskeisten analyysin osien kopiointi hiirellä johonkin tekstinkäsittelyohjelmaan.

Grafiikkatiedostojen tulostaminen onnistuu komennolla `dev.print` ja tallentaminen komennoilla `dev.copy2eps` tai `dev2bitmap` — Windowsissa myös graafisesti suoraan ikkunavalikosta. Komento `dev.copy2eps` tuottaa `Encapsulated PostScript` -tiedostoja, jotka ovat julkaisutarkoituksiin yleensä parhaita: ne käyttävät vektorigrafiikkaa, joten ne ovat skaalattavia ja tarkkoja. Windows-ohjelmilla tosin näyttää olevan usein ongelmia niiden käsittelyssä. Komento `dev2bitmap` tuntee monia vaihtoehtoisia bittikarttaformaatteja, jotka sopivat esim. verkkokäyttöön, mutta ovat yleensä huonolaatuisia käsikirjoituksissa ja julkaisuissa. Hieman epäloogisesti `dev2bitmap` tuottaa myös vektorigrafiikkaa käyttäviä `pdf`-tiedostoja, joskin niiden formatointi poikkeaa hieman ruudulla näkyvästä ja muiden vaihtoehtojen tallentamasta kuvasta.

2.8 R:n mukauttaminen (*customizing*)

R:n ulkoasua ja käyttäytymistä voi säädellä optioilla. Esimerkiksi minä suhtaudun ylenkatseella "merkitsevyystähtiin" ja tämän oppaan esimerkeissä niitä ei näy. Olen saanut tämän aikaan komennolla:

```
> options(show.signif.stars=FALSE)
```

joka ylittää asennuksen oletuksen tulostaa tähdet. R:n mukauttamiskomentoja ovat:

- `options`: yleiset optiot.
- `par`: graafiset optiot.
- `ps.options`: Postscript–optiot, jotka vaikuttavat kuvien tulostukseen ja kuvatiedostoihin.

Optioita on hyvin paljon, ja ne saattavat vaihtua versioittain: `help` antaa ajantasaisen tiedon. Optioiden vallitsevat arvot näkee ehkä tiiveimmin komennolla

```
> str(options())
```

R:n asennuksen myötä systeemissä on tiedosto `Rprofile`, joka sisältää oletusasetukset (ja jota R:n ylläpitäjä on voinut muuttaa). Jokainen käyttäjä voi kirjoittaa itselleen oman piilotiedoston `.Rprofile`, joka syrjäyttää systeeminlaajuiset asetukset. R etsii tätä tiedostoa ensin työhakemistosta ja sitten käyttäjän kotihakemistosta eli käyttäjällä voi olla erilaisia `.Rprofile`ja, mutta vain ensin löydetty toimii. Lisäksi käyttäjä voi kirjoittaa työhakemistossa toimivan funktion `.First`, joka suoritetaan aloitettaessa istunto (katso `help(.First)`). Tässä voi olla optioiden lisäksi muitakin komentoja, esim. tietyt kirjastot voidaan automaattisesti lukea istuntoon.

3 Käytön perusteet

3.1 Paisunut laskin

R on interaktiivinen, komentopohjainen ohjelma. Siinä on erinomaiset matemaattiset ja tilastolliset perusvalmiudet, joten se toimii mainiosti myös laskimena. Matemaattinen lauseke kirjoitetaan R-kehotteeseen, ja R antaa välittömästi vastauksen:

```
> log(pi)/sqrt(2)
[1] 0.8094463
```

R tuntee matemaattiset operaattorit `+`, `-`, `*`, `/` ja `^` (potenssiin korotus). Lisäksi se tuntee tavalliset matemaattiset funktiot kuten `log` (luonnollinen logaritmi tai minkä tahansa kannan `base` logaritmi), `log10`, `log2`, `exp`, `sin`, `tan`, `sqrt` jne.

Laskujen tulokset voi tallettaa muuttujiin sen sijaan että ne tulostaisi suoraan näytölle. R:n ensisijainen sijoitusoperaattori "`<-`" koostuu kahdesta peräkkäisestä merkistä "`<`" eli "pienempi kuin" ja "`-`" eli "tavuviiva". Sijoitusoperaattoriksi kelpaa myös "`=`" eli "on yhtä kuin" — ainakin yleensä.[3] Talletetun muuttujan arvon voi tulostaa näytölle kirjoittamalla muuttujan nimen, mikä epäsuorasti kutsuu komentoa `print`:

```
> r <- 2
> A <- pi*r^2
> A
[1] 12.56637
```

Huomattakoon, että R laskee tarkemmin kuin tulostaa — tulostettavien numeroiden määrää voi muuttaa (`options(digits)`).

R-muuttujien nimissä saa olla merkkejä `A-Za-z0-9` ja `.`, mutta nimi ei saa alkaa numerolla eikä pisteellä. Etenkin on huomattava, että matemaattiset symbolit (`+`, `-`, `:`, `*`, `/`) ja alaviiva "`_`" *eivät ole sallittuja* vaan voivat johtaa hämmentäviin virheisiin (alaviiva siksi, että se on sijoitusoperaattori). Itse asiassa joissain R-asennuksissa saattavat kelvata myös suomalaiset ääkköset, mutta niiden käyttökelpoisuus on riippuvaista koneen asetuksista ja jopa käyttöjärjestelmästä, joten niitä on järkevintä olla käyttämättä vaikka ne tänään ja täällä sattuisivatkin kelpaamaan (huomenna ja toisaalla kaikki voi olla toisin). Samaten tyhjät välit ("blankot") eivät ole sallittuja muuttujien nimissä. Näitä määräyksiä pystyy usein kiertämään laittamalla nimet lainausmerkkien (`"`) väliin, mutta tästä seuraa melko varmasti vaikeaselkoisia ongelmia — ennemmin tai myöhemmin.

3.2 Vektorit

R on tilastollinen ohjelma, joten yleensä ei riitä, että se operoi vain yksittäisillä luvuilla. Perusrakenne on *vektori*, jota käytetään etenkin *muuttujien* esittämiseen. Vektorit syntyvät usein luettaessa havaintoaineisto R:ään, mutta niitä voi muodostaa myös R-istunnossa.

Perusfunktio vektorien muodostamiseksi on `c`, joka yhdistää (engl. *"concatenate"*) arvoja vektoriksi. Pieniä havaintoaineistoja voi tallettaa suoraan tällä tavoin:

```
> vek <- c(3.4, 2, 2.22, 0.9, 5.12)
> vek
[1] 3.40 2.00 2.22 0.90 5.12
```

Vektoreilla on ainakin kaksi ominaisuutta: pituus eli alkioiden määrä (`length`) ja alkioiden tyyppi (`mode`):

[3] Sijoitusoperaattoksi kelpaa toistaiseksi myös "`_`" eli alaviiva, muttemme suosittele sen käyttöä, koska se tekee koodista vaikealukuisen. Se tulee myös poistumaan tulevista R-versioista ja tällä hetkellä R jo varoittaa sen käyttäjää.

```
> length(vek)
[1] 5
> mode(vek)
[1] "numeric"
> length(A)
[1] 1
```

Vektorissa `vek` on 5 numeerista alkiota. Yksittäiseen vektorin alkioon viitataan hakasulkuihin `[]` sijoitetulla indeksillä. Matemaattista merkintää x_i vastaa R:ssä `x[i]`. Itse asiassa R piti myös aiemmin muodostamaamme vakiota `A` yhden alkion mittaisena vektorina. Saamme siis vektorin `vek` kolmannen alkion (2.22) tai muuttujan `A` ainoan alkion arvot kirjoittamalla:

```
> vek[3]
[1] 2.22
> A[1]
[1] 12.56637
```

Myös vektorin indeksi voi olla vektori, jolloin saamme kaikki vektorissa luetellut alkiot. Esim. alkiot 3 ja 1 tässä järjestyksessä ovat:

```
> vek[c(3,1)]
[1] 2.22 3.40
```

Kaikkien vektorin alkioiden on oltava samaa tyyppiä, mutta tyyppejä on tarjolla useita. Erityisen hyödyllisiä tilastollisissa analyyseissä ovat tekstinä ilmaistut luokkamuuttujat:

```
> pihalla <- c("helsinki","tampere","jyväskylä","oulu")
> pihalla
[1] "helsinki"  "tampere"   "jyväskylä" "oulu"
> mode(pihalla)
[1] "character"
> length(pihalla)
[1] 4
```

Tässä tapauksessa muuttujat ovat merkkijonoja (`character`). R:ssä myös tällaisia merkkijonomuuttujia voi käyttää luokittelevina muuttujina analyyseissä. Tällöin niitä sanotaan *faktoreiksi*. Sekä numeerisen että merkkijonomuuttujan voi muuntaa faktoriksi komennolla `factor`:

```
> pihalla <- factor(pihalla)
```

Monessa tilasto-ohjelmassa luokkamuuttujatkin on koodattava numeroiksi, mutta tekstinä esittäminen tekee tulokset *paljon* helpommin tulkittaviksi — varsinkin aikojen kuluttua. Palaamme faktoreihin perusteellisemmin varianssianalyysin yhteydessä.

Kolmas tavallinen tyyppi on looginen muuttuja (`logical`), joka voi saada vain arvoja tosi (`TRUE`) tai epätosi (`FALSE`). Looginen muuttuja syntyy yleensä ehtolauseiden tuloksena:

```
> vek > 2
[1]  TRUE FALSE  TRUE FALSE  TRUE
```

Yllä käytettiin vertailuoperaattoria `>`. Mikäli olisimme halunneet myös arvon 2 olevan tosi, olisimme käyttäneet ehtoa "`>=`". Muita vertailuoperaattoreita ovat `<`, `<=`, `==` (yhtäläisyys), `!=` (epäyhtäläisyys) ja `!` (negaatio). Loogisia muuttujia joutuu käyttämään kun kutsuttavalle funktiolle on kerrottava, onko tietyn parametrin arvo tosi vai epätosi; tällöin ne saa lyhentää muotoon `T` ja `F`.

Loogisia operaattoreita käytetään usein valitsemaan osa-aineistoja: kun vektorin indeksinä on looginen vektori, valitaan ne alkiot, joille ehto on tosi. Voimme valita suuremmat arvot kuin 2 kirjoittamalla:

```
> vek[vek>2]
[1] 3.40 2.22 5.12
```

Hakasuluissa olevan ehtolauseen tulos on vektori, joka on `TRUE` alkioille 1, 3 ja 5.

3.3 Laskutoimitukset vektoreilla

Vektoreita voi käyttää matemaattisissa lausekkeissa melko samoin kuin yksittäisiä lukujakin. Mikäli jokin laskutoimituksen tekijä on vektori, kierrätetään lyhyempiä tekijöitä tarpeeksi monta kertaa ja tulos on pisimmän vektorin pituinen (mikäli tämä tehdään tarkoituksella eikä vahingossa, vaaditaan yleensä tarkkaa harkintaa eripituisia vektoreita yhdistettäessä). Tutulla vektorillamme `vek` ja muuttujallamme `A` saamme:

```
> A*vek
[1] 42.72566 25.13274 27.89734 11.30973 64.33982
```

Mikäli vektorit ovat samanmittaiset, kaikki matemaattiset operaatiot tehdään *alkioittain*:

```
> mult <- floor(vek)
> mult
[1] 3 2 2 0 5
> mult*vek
[1] 10.20  4.00  4.44  0.00 25.60
```

Useimmille käyttäjille on helpotus, että R ei tällaisessa tapauksessa suorita matriisialgebran vektorikertolaskua. Tosin sitä haluaville on toki tarjolla myös matriisikertolasku (R-intro, 2000).

Vektoreille on koko joukko erityisiä funktioita: `min` ja `max` palauttavat vektorin pienimmän ja suurimman arvon, `sum` vektorin arvojen summan, `prod` arvojen tulon, `mean` keskiarvon, `var` varianssin, `sd` hajonnan (`length` olikin jo mainittu). Funktio `range(`*`vektori`*`)` palauttaa kahden arvon mittaisen vektorin `c(min(`*`vektori`*`), max(`*`vektori`*`))`.

Joskus on tarve käyttää poikkeamia keskiarvosta ($\bar{x}$) tai tehdä ns. z-muunnos

$$z_i = \frac{x_i - \bar{x}}{s} \tag{1}$$

eli vähentää kustakin alkiosta keskiarvo ja jakaa keskihajonnalla, jonka jälkeen muunnettujen lukujen z keskiarvo on $\bar{z} = 0$ ja hajonta on $s = 1$. Tämän voi tehdä komennolla

```
>  scale(x)          ## z-muunnos
>  scale(x, scale=F) ## vain keskiarvon vähentäminen
```

Vektorin arvot voi lajitella komennolla `sort`. Komennot `sort.list` ja `order` palauttavat indeksit, jotka järjestäisivät vektorin. Näitä epäsuoria komentoja käytetään silloin, kun halutaan järjestää monta vektoria samaan järjestykseen yhden vektorin mukaan:

```
> sija <- c(1,2,3,4,5)  ## Oikotie ■3.4
> sort(vek)
[1] 0.90 2.00 2.22 3.40 5.12
> i <- sort.list(vek)   ## tai order(vek)
> vek[i]
[1] 0.90 2.00 2.22 3.40 5.12
> sija[i]
[1] 4 2 3 1 5
```

3.4 Säännölliset sarjat

Sangen usein tarvitaan vektoreita, joiden arvot ovat tasavälisiä. Kokonaislukusarjan saa komennolla ``*alku*`:`*loppu*``:

```
> 3:12
 [1]  3  4  5  6  7  8  9 10 11 12
```

Tällaista sarjaa tarvitaan, kun halutaan nähdä vain osa vektorin arvoista tai tallettaa vain osa toiseen vektoriin. Meillä ei tosin ole vielä pitkää vektoria, mutta esimerkki riittänee:

```
> vek[2:4]
[1] 2.00 2.22 0.90
```

Kaksoispiste (`:`) on itse asiassa lyhenne komennolle `seq`, jonka avulla voidaan tuottaa myös muita kuin perättäisten kokonaislukujen sekvenssejä. Komentoa voi kutsua monella tavalla:

```
>  from:to
>  seq(from, to)
>  seq(from, to, by=)
>  seq(from, to, length=)
>  seq(along)
```

Parametrit `from` ja `to` voi ilmoittaa suoraan numeroina (sillä ne ovat kaksi ensimmäistä), mutta vaihtoehtoinen kolmas parametri vaatii nimen. Parametri `by` ilmoittaan lisäyksen (joka voi olla negatiivinen alenevissa sarjoissa) tai vaihtoehtoisesti `length` ilmoittaa muodostettavien alkioiden määrän. Lopulta parametri `along` kertoo, että muodostetaan jonkin toisen vektorin pituinen kokonaislukusekvenssi:

```
> 1:6
[1] 1 2 3 4 5 6
> seq(1,6)/5
[1] 0.2 0.4 0.6 0.8 1.0 1.2
> seq(0,4,by=0.6)
[1] 0.0 0.6 1.2 1.8 2.4 3.0 3.6
> seq(0,4,length=7)
[1] 0.0000000 0.6666667 1.3333333 2.0000000 2.6666667 3.3333333 4.0000000
> seq(along=vek)  ## Tässä luvattu oikotie
[1] 1 2 3 4 5
```

Toinen hyödyllinen funktio säännöllisten sarjojen muodostamiseksi on `rep`, joka toistaa samaa arvoa. Saattaa kuulostaa turhalta, mutta on erittäin hyödyllistä esimerkiksi faktorien muodostamisessa. Yksinkertaisimmassa muodossaan funktiota kutsutaan:

```
> c(rep("contr",3), rep("treat",3))
[1] "contr" "contr" "contr" "treat" "treat" "treat"
```

3.5 Puuttuvat tiedot

Periaatteessa kaikkien vektorin muuttujien on oltava samaa tyyppiä. Jokin arvoista voi kuitenkin puuttua esimerkiksi tutkimuksellisen ongelman takia. Puuttuva tietoa merkitään R:ssä `NA` (“*Not Available*”). Toinen syy tiedon puuttumiseen on mahdoton matemaattinen operaatio:

```
> v2 <- -1:3
> v2
[1] -1  0  1  2  3
> log(v2)
[1]       NaN      -Inf 0.0000000 0.6931472 1.0986123
Warning message:
NaNs produced in: log(x)
```

Mahdottoman matemaattisen operaation tuloksena syntyvä puuttuva tieto on yleensä `NaN` (*“Not a Number”*), mutta R pystyy myös tuottamaan tulokseksi äärettömän (`Inf`), kuten yllä näkyy.

Puuttuvat tiedot ovat riesa analyysissä, sillä jos laskutoimituksessa on puuttuvat tieto, on tulos myös puuttuva tieto:

```
> mean(sqrt(v2))
```

```
[1] NaN
```

Ainoa mitä puuttuville tiedoille rehellisesti pystyy tekemään, on poistaa havainnot, joissa on puuttuvia tietoja. Monissa vektorioperaatioissa on tätä varten optio `na.rm`:

```
> mean(sqrt(v2), na.rm=T)
[1] 1.036566
```

Tämä ei olisi auttanut `log`-funktiolle, sillä siinä tuloksena oli myös `-Inf`, jota R ei pitänyt puuttuvana tietona. Sen sijaan sekä `NaN` että `NA` käsitellään puuttuvina tietoina. Käyttäjän kannalta on usein yhdentekevää, onko tieto `NaN` vai `Inf`, sillä harvoin halutaan tulosta:

```
> logv2 <- log(v2)
Warning message:
NaNs produced in: log(x)
> mean(logv2, na.rm=T)
[1] -Inf
```

Vaikka `NaN` poistettiin, jäi `-Inf` jäljelle, joten keskiarvo oli `-Inf`. Tämä on joskus riesa, mutta on myös laskutoimituksia, jotka ovat määriteltyjä äärettömille suureille:

```
> exp(logv2)            ## exp(-Inf) = 0
[1] NaN   0   1   2   3
```

Looginen funktio `is.na` on tosi, jos alkio on puuttuva tieto. Looginen funktio `is.finite` taas on tosi jos alkio on äärellinen. Voimme siis poistaa puuttuvat tiedot (`NA` ja `NaN`) tai ei-äärelliset alkiot kirjoittamalla:

```
> logv2[!is.na(logv2)]  ## "!" on looginen negaatio
[1]      -Inf 0.0000000 0.6931472 1.0986123
> logv2[is.finite(logv2)]
[1] 0.0000000 0.6931472 1.0986123
> mean(logv2[is.finite(logv2)])
[1] 0.5972532
```

Laajemmin puuttuvien tietojen käsittelyyn käytetään funktioita `na.action`, `na.omit` ja `na.fail`.

3.6 Matriisit

R tuntee myös matriisit, jotka ovat kaksi- tai useampiulotteisia taulukoita. Matriisi voidaan muodostaa keskenään samanmittaisista vektoreista. Komento `cbind` yhdistää vektorit siten, että kukin tulee matriisin sarakkeeksi ("*column*"). Komento `rbind` taas tekee yhdistämisen toisin päin, eli kukin vektori tulee omaksi rivikseen ("*row*").

```
> luvut <- 2:5
> neliot <- luvut^2
> cbind(luvut,neliot)
     luvut neliot
[1,]     2      4
[2,]     3      9
[3,]     4     16
[4,]     5     25
> rbind(luvut,neliot)
       [,1] [,2] [,3] [,4]
luvut     2    3    4    5
neliot    4    9   16   25
```

Matriisilla on ainakin kaksi dimensiota ja vastaavasti useampi indeksi. Matemaattisesti merkitsemme x_{ij} ja R:ssä `x[i,j]`. Kuten näkyy, dimensiolla voi olla myös nimi, jota voi käyttää indeksinä. Niinpä seuraavat tavat ovat vaihtoehtoisia:

```
> taulu <- cbind(luvut,neliot)
> taulu[,"neliot"]
[1]  4  9 16 25
> taulu[,2]
[1]  4  9 16 25
```

Yllä olevissa esimerkeissä ensimmäinen indeksi jätettiin pois (pilkkua ennen ei mitään), joten kaikki sen arvot tulostettiin. Aivan samoin kuin vektoreissa, indeksi voi olla vektori. Esimerkiksi `taulu[1:3,1]` tulostaisi ensimmäisen sarakkeen kolme ensimmäistä riviä. Matriisissa voi olla myös useampi kuin kaksi dimensiota. Tällöin indeksejä on tietysti useampi. Matriisin dimensiot näkee komennolla `dim`:

```
> dim(taulu)
[1] 4 2
```

Matriiseja käsitellään laskutoimituksissa yleensä kuten vektoreita eli kaikki operaatiot tehdään alkioittain. Halullisille on toki jälleen tarjolla varsinaisia matriisioperaatioita, mutta niistä enemmän pitemmälle menevissä oppaissa.

3.7 Havaintoaineistot (*data.frame*)

Tavanomaiset havaintoaineistot (*data set*) tai havaintomatriisit (*data matrix*) ovat myös eräänlaisia matriiseja eli kaksiulotteisia taulukoita. Havaintomatriisin sarakkeita sanotaan *muuttujiksi* ja rivejä *havainnoiksi*. Tilastollisen havaintoaineiston muuttujat eivät kaikki välttämättä ole numeerisia eli reaaliarvoisia (`numeric`), vaan luokitteluasteikon muuttujat voivat olla tyypiltään merkkimuotoisia (`character`) tai loogisia (`logical`). Tilastollista havaintoaineistoa kuvaavaa kaksiulotteista taulukkoa varten R:ssä onkin nimetty erityinen datarakenne- tai oliotyyppi (ks. s. 15) nimeltä `data.frame` eli "*datakehikko*". Valmis havaintoaineisto luetaan datakehikoksi yleensä ulkoisesta tiedostosta (§3.9)). Lisäksi R:ssä on valmiina suuri joukko esimerkkiaineistoja, jotka saa käyttöön komennolla `data(aineiston.nimi)`. Tämän oppaan liitepaketissa `rekola` on pikkuruinen aineisto `ih98`, jossa on Helsingin yliopiston ympäristöekologian laitoksen ekologian laboratoriokurssilla kerätty pieni aineisto, joissa on ruukuissa kasvatettujen retiisien lukumäärä per ruukku (`tiheys`) sekä kuivapaino (`massa`). Tämän aineiston saamme käyttöön komennoilla:

```
>  library(rekola)
>  data(ih98)
```

Voimme nyt viitata tämän `data.frame`na käsiteltävän havaintomatriisin `ih98` alkioihin, riveihin ja sarakkeisiin normaalisti indekseillä:

```
> dim(ih98)
[1] 25  2
> ih98[1:3,]
  tiheys  massa
1      1 0.5156
2      2 0.2617
3      4 0.4769
```

Voimme käyttää indekseinä myös muuttujien nimiä. Emme kuitenkaan voi suoraan käyttää `data.frame`n sisäisiä muuttujien nimiä `tiheys` ja `massa` vaan meidän on kirjoitettava `ih98$tiheys` ja `ih98$massa`, mikä on vähintäänkin kömpelöä. Voimme kuitenkin *liittää* (`attach`) tämän datakehikon istuntoon, minkä jälkeen aineiston muuttujat ovat suoraan viitattavissa:

```
> massa[1:5]
Error: Object "massa" not found
> ih98$massa[1:5]
[1] 0.5156 0.2617 0.4769 0.1913 0.3135
> attach(ih98)
> massa[1:5]
```

```
[1] 0.5156 0.2617 0.4769 0.1913 0.3135
```

Tällöin meidän on tietenkin syytä varoa, ettei meillä ennestään ole samannimisiä muuttujia (esim. `x` näyttää olevan kovin yleinen nimi).

Kun emme enää tarvitse aineistoa, voimme irrottaa sen

```
> detach(ih98)
```

minkä jälkeen muuttujat `tiheys` ja `massa` eivät enää suoraan näy R-istuntoon.

3.8 Luettelot (*lists*)

Luettelot eli listat (`list`) ovat vielä yksi R:n datatyypeistä. Luettelossa voi olla erimittaisia ja erinimisiä nimettyjä komponentteja. Peruskäyttäjä joutuu harvemmin luomaan luetteloita, mutta monet ohjelmat tuottavat tuloksenaan luettelon. Esimerkiksi regressioanalyysin tulos funktiosta `lm` on luettelo (§7). Siinä on nimettyinä komponentteina mm. sovitetut arvot, residuaalit, jäännösvaihtelu, regressiokertoimet, vapausasteet, malliyhtälö ja paljon muuta. Näihin komponentteihin viitataan kuten aineiston muuttujiin: `luettelo$komponentti`. Menettelyn etuna on, että kaikki mallin tulokset kulkevat yhdessä roskaamatta istuntoa lukemattomalla määrällä analyysin tuottamia muuttujia, joita käyttäjä ei ehkä koskaan tarvitse.

Itse asiassa aineisto on luettelon erikoistapaus: kaikki komponentit ovat samanmittaisia vektoreita eli havaintojen määrä on sama kaikille muuttujille.

Luettelo muodostetaan komennolla `list`, jonka parametrilistassa esitellään luettelon komponentit ja muuttujat, joista ne muodostetaan:

```
> lista <- list(komp1 = x, komp2 = y)
```

3.9 Ulkoisen tiedoston lukeminen aineistoksi

Suuremmat aineistot luetaan yleensä R:ään ulkoisista tiedostoista. Paketissa `foreign` on funktioita eräiden tilasto-ohjelmien tiedostojen lukemiseksi R:ään. Usein on kuitenkin varminta käyttää `read.table` -funktiota. R pystyy varmuudella lukemaan ascii-tiedostoja, eli sellaisia tiedostoja, joissa on vain tavallisia näkyviä merkkejä. Yksittäisten arvojen on oltava jollain tavoin erotettuja: esimerkiksi tyhjä väli on kelvollinen. Funktio `read.table` hyväksyy monenlaisia erottimia, mutta saattaa olla tarpeen ilmoittaa funktiolle käytetty erotin (`sep`). Muuttujien ja havaintojen nimet voidaan myös lukea tiedostosta.

Funktion `read.table` toiminta tuntuu vaihtelevan hieman R-versioissa, mutta ainakin joissan versioissa tarjolla on seuraavat vaihtoehdot tiedoston lukemiseksi:

- Ensimmäisellä rivillä on muuttujien nimet ja ensimmäisellä sarakkeella havaintojen nimet. Tällöin ensimmäisellä rivillä on yksi alkio vähemmän kuin muilla.

  ```
  > aineisto <- read.table("tiedosto")
  ```

- Ensimmäisellä rivillä on muuttujien nimet, mutta havainnoilla ei ole nimiä. Kaikilla sarakkeilla on siis yhtä monta alkiota. Havainnot numeroidaan aineistossa.

  ```
  > aineisto <- read.table("tiedosto", header=T)
  ```

- Sekä muuttujat että havainnot ovat nimeämättömiä, eli alkioita on kaikilla riveillä yhtä monta. Havainnot numeroidaan ja muuttujat nimetään esim. `V1`, `V2`,...

  ```
  > aineisto <- read.table("tiedosto")
  ```

Sekä muuttujien että havaintojen nimet pystyy antamaan tai muuttamaan myös myöhemmin (`colnames`, `rownames`).

Usein aineisto on peräisin jostain taulukkolaskentaohjelmasta, esim. Excelistä. Taulukkolaskentaohjelmien yksityisiä formaatteja R ei pysty lukemaan (ainakaan kaikilla alustoilla). Tällainen tiedosto on ensin muutettava johonkin yllä mainituista taulukkotyypeistä. Taulukkolaskentaohjelmien `csv` ("*Comma Separated Values*") eli pilkkujen erottamat arvot on käyttökelpoinen tallennusmuoto. `Csv`-formaattia varten R:ssä on kaksi `read.table` -funktion varianttia: `read.csv` ja `read.csv2`. Jälkimmäinen on tarkoitettu suomalaisilla asetuksilla toimiville ohjelmille, jotka käyttävät desimaalierottimena pilkkua normaalin pisteen sijaan ja vastaavasti lukujen erottimena puolipistettä pilkun sijaan.

Taulukkolaskentatiedostoja talletettaessa on syytä tarkastaa tiedosto huolellisesti. R:ää varten on talletettava vain varsinainen taulukko, ei erilaisia lisälaskelmia ja kuvia. Erityisen huolellisesti on tarkastettava, että muuttujien ja havaintojen nimet noudattavat R:n sääntöjä: ei välilyöntejä, alaviivoja ja varmuuden vuoksi ei myöskään ääkkösiä. Sanat on siis kirjoitettava yhteen ja tarpeen vaatiessa käytettävä nimissä pistettä välien tai alaviivojen sijaan. Merkkilajisokeiden käyttöjärjestelmien (Windows) ja merkkilajiherkän R:n yhteistoiminta voi myös vaatia huomiota.

Tiedostoissa voi olla myös puuttuvia tietoja. Lukukomennoille voi ilmoittaa puuttuvista tiedoista käytetyn symbolin (`na.string=`). Mikäli erottimena käytetään jotain muuta merkkiä kuin tyhjää (esim. `read.csv`), tyhjät kentät voidaan lukea suoraan puuttuviksi tiedoiksi. Ainakin Excelin käyttäjillä on usein tapana jättää nollat kirjoittamatta jolloin R saattaa tulkita ne puuttuviksi tiedoiksi. Olen tosin useammin kuin harvoin nähnyt, että käyttäjät tekevät tämän virheen huomaamattaan jo Excelissä. Tyhjät kentät voi tulkita myös nolliksi tai muuttaa ne myöhemmin nolliksi

```
> x[is.na(x)] <- 0
```

mutta tällöin olisi parasta että *kaikki* muuttujan samoin koodatut kentät ovat joko nollia tai puuttuvia tietoja.

Mikäli kaikki muuttujan arvot ovat R:n mielestä numeerisia, muuttuja tulkitaan numeeriseksi. Mikäli R on mielestään löytänyt muuttujasta yhdenkin ei-numeerisen arvon, muuttuja tulkitaan faktoriksi. Jos siis numeeriseksi kuviteltu muuttuja onkin R:n mielestä faktori, on syytä tarkistaa tiedosto ja komentonsa. Muuan helpoimmista tavoista tarkistaa aineiston lukeminen on komento

```
> str(aineisto)
```

joka tulostaa kunkin muuttujan ensimmäiset havainnot ja näyttää tyypin. R:ään pystyy lukemaan myös tekstimuuttujia faktoreiksi. Tosin monet taulukkolaskentaohjelmat (Excel) eivät pysty itse käsittelemään merkkimuuttujia kovinkaan hyvin, joten faktoritkin on usein epäinformatiivisesti koodattu numeroiksi. Kannattaa vakavasti harkita tällaisten muuttujien koodaamista selväkielisiksi nimiksi. Tavallinen syy numeerisen muuttujan "faktoristumiseen" on, että muuttujassa oli puuttuvia tietoja, joita ei ollut asiallisesti koodattu. Parametri `as.is` voi auttaa tällaisten tilanteiden hallinnassa.

Aineiston lukemisen jälkeen on vielä mahdollista muuttaa muuttujien tyyppiä. R:ssä on koko joukko `as.xxx` -muunnosfunktioita, missä `xxx` on jokin R:n lukuisista tyypeistä. Alkujaan numeroina koodattu ja numeeriseksi tulkittu faktori voidaan todella muuttaa faktoriksi myös suoralla komennolla:

```
> aineisto$a <- factor(aineisto$a, levels=c(...))
```

Parametrilla `levels` voi luetella faktorin tasot, jolloin ne tulevat käyttäjän haluamaan järjestykseen R:n suosiman sijaan. Pysyvien muutosten aikaan saamiseksi on usein käytettävä tätä pitempää luettelotyyppistä muotoa muuttujasta `a` vaikka aineisto olisi liitetty (`attach`) R:ään, sillä muuten muuttuu vain paikallinen kopio aineistosta (sama koskee tietysti puuttuvien tietojen korvaamista yllä).

3.10 Ohjelmointi

R on täysiverinen ohjelmointiympäristö, aivan samoin kuin serkkunsa S. R:ssä on ohjelmointikielelle välttämättömät ehtorakenteet (`if`, `ifelse`, `switch`) ja toistorakenteet (`for`, `while`, `repeat`). Monessa R- ja S-oppaassa pääpaino onkin ohjelmoinnissa (esim. R-intro, 2000). Ei kuitenkaan tässä oppaassa. Esitän vain yhden yksinkertaisen näytteen oman funktion kirjoittamisesta — lisäohjeita ja esimerkkejä löytyy sekä muista oppaista että katsomalla, miten R-funktiot on kirjoitettu. Suuri osa R:n komennoista on näet kirjoitettu R-kielellä ja käyttäjän itse kirjoittama funktio on aivan samassa asemassa ja yhtä tehokas kuin moni R-komento. Kirjoittamalla komentonimen ilman sulkuja, R tulostaa komennon näytölle ihmeteltäväksi.

Esimerkkimme on hyvin lyhyt, mutta voimakas. Kirjoitan pienen funktion, joka simuloi lajin havaittuja runsauksia ekologisella gradientilla. Olettakaamme, että lajin odotettu runsaus μ gradientin arvolla x määräytyy gaussilaisen responssimallin mukaan:

$$\mu = h \exp\left[-\frac{(x-u)^2}{2t^2}\right] \tag{2}$$

missä lajin responssifunktion parametrit ovat lajin optimin sijainti gradientilla (u), lajin odotettu runsaus optimissa (h) sekä lajin responssin leveyttä kuvaava parametri (t). Todelliset havainnot eivät kuitenkaan ole virheettömiä, joten meidän on simuloitava satunnaisuutta. R:ssä on suuri joukko funktioita, jotka generoivat satunnaislukuja erilaisista jakaumista. Tässä esimerkissä simuloimme havaittuja runsauksia, joten tarvitsemme Poisson-jakautuneita satunnaislukuja eli funktiota `rpois`. Muita usein tarvittuja vaihtoehtoja ovat tasainen jakauma (`runif`) sekä normaalijakauma (`rnorm`), mutta vaihtoehtoja on paljon lisää. Gaussilainen mallimme antaa odotusarvon μ, jota käytämme Poisson-jakauman parametrina ja tuotamme satunnaislukuja y eli formaalisemmin $y \sim P(\mu)$.

Meidän tarvitsee kirjoittaa vain pari riviä:

```
>  gausresp <- function(x, h=1, u=0, t=1) {
    mu <-  h*exp(-(x-u)^2/2/t^2) # Yhtälö 2
    y <- rpois(length(mu), mu)   # Poisson satunnaisluku
    y
  }
```

Funktiolla on neljä parametria, joista vain gradientin arvo (`x`) on pakollinen ja muille on oletusarvot.

Ensi katsomalta funktio näyttää melko hyödyttömältä, sillä se näyttää hyväksyvän vain yhden gradienttiarvon. Muistakaamme kuitenkin, että jos laskutoimituksen tekijä on vektori, muita tekijöitä kierrätetään eli voimme kutsua funktiota `x`-vektorilla:

```
> gausresp(seq(4.5,6.9,length=18), u=5.3, h=8.2)
 [1]  4  6  5  7  4 10  4  3 10  9  7  7  5  6  5  3  2  4
```

Koska lajin parametrit olivat nimettyjä, saimme antaa ne mielivaltaisessa järjestyksessä eikä meidän tarvinnut antaa parametria `t`, koska käytimme sen odotusarvoa.

Monessa tilastollisessa ohjelmassa on makroja, mutta R:ssä on funktioita. Makrot ovat eräänlaisia lyhenteitä komennoille, jotta ei tarvitsisi aina kirjoittaa pitkiä, toistuvia komentoketjuja. Suoritettaessa makro laajenee normaaliin istuntoon. Jos meillä olisi ollut ennestään muuttujat `mu` tai `y`, makro olisi antanut näille uudet arvot ja alkuperäiset olisivat tuhoutuneet. Funktiolla sen sijaan on yksityinen nimiavaruus, eli voimme turvallisesti käyttää funktiossa samoja nimiä kuin istunnossa. Toisaalta funktio ei palauta mitään ellemme erityisesti pyydä: tämän takia yksinäinen "`y`" viimeisellä rivillä palauttamaan tulokset kutsuvalle ohjelmalle. Sen sijaan muuttujan `mu` arvoja emme koskaan saa tietää kutsuvassa ohjelmassa. Perustellusti voisi tietysti vaatia, että

saisimme tiedon myös `mu`:n arvoista, tai ainakin käytetyistä `x`:n arvoista sekä lajin parametreista. Tämä olisi voitu hoitaa palauttamalla luettelo (luku 3.8) yksittäisen `y`-vektorin sijaan. Tämä olisi tapahtunut korvaamalla viimeinen rivi "`y`" riveillä:

```
out <- list(gradient=x, simulated=y, h, u, t)
out
```

Virallista luettelomuotoa (§3.8) tarvittiin vain kun haluttiin antaa muuttujalle uusi nimi luettelossa. Jo tällaisenaan funktiomme on etenkin virhejakauman suhteen parempi kuin monet suositut simulointiohjelmat ja helposti kehitettävissä monimutkaisempiin tilanteisiin.

3.11 Oliot

R:ssä on olioita, joilla on periytyviä ominaisuuksia. Yksinkertaisimpia kohtaamistamme olioista on `data.frame` (kpl. 3.7). Olion *luokan* saa tietää komennolla `class`. Monet funktiot, esimerkiksi regressioanalyysi, palauttavat luettelon, jossa on tavattoman suuri määrä komponentteja. Näiden joukossa ovat esimerkiksi sovitetut arvot, residuaalit, regressiokertoimet, vapausasteet ja paljon muuta. Regressionalyysin tulos on olio, joka kuuluu luokkaan `lm` (luku 7). Kun talletamme regressioanalyysin tuloksena syntyvän olion, meillä on tallessa kaikki analyysin kannalta tärkeä informaatio myös myöhempään käyttöön. Monet menetelmät tietävät, kuinka esittää tiettyyn luokkaan kuuluva olio. Esimerkiksi komento `print` (jota usein kutsutaan epäsuorasti kirjoittamalla olion nimi) ei tulosta kaikkia regressioanalyysin tuloksen osia, vaan ainoastaan luokalle `lm` määritellyt keskeiset tiedot. Kirjoittamalla

```
> methods(print)
```

käy ilmi, että yhden `print`-komennon sijaan R:ssä onkin suuri joukko komentoja `print.`*`luokka`*, ja esim. luokalle `lm` käytetty komento on itse asiassa `print.lm`. Pisteen jälkeistä osaa ei tarvitse kirjoittaa, vaan oikean luokan menetelmää kutsutaan automaattisesti. Muita monelle luokalle saatavilla olevia komentoja ("menetelmiä") ovat esim. `summary` ja `plot`. Niinpä pelkällä `plot`-komennolla piiretyt kuvat on sopeutettu kyseiseen olioon. Mikäli haluamme paremman kuvan siitä, mitä olio on todella syönyt, voimme käyttää komentoja

```
> attributes(olio)
```

```
> str(olio)
```

4 Grafiikan perusteet

Tämä kappale ei ole runsaasti kuvitettu. Tarkoitus on vain lyhyesti esitellä tärkeimmät grafiikkakomennot, joita tullaan myöhemmin käyttämään. Itse asiassa muuallakin monisteessa olen säästellyt kuvissa: tarkoitus on, että lukija tekee kuvat itse R-istunnossa. Jätän esittelemättä sekä kuvat, joita en pidä kovin hyväksyttävinä (esim. `pie`) sekä monimutkaiset kuvat, joita tässä oppaassa ei käytetä (esim. kolmiulotteiset kuvat, aikasarjat). Myös monen analyysiohjelman tuloksille on omat oletuskuvat tai kuvastonsa (esim. lineaarisen mallin tulokselle kuva 6).

4.1 `plot`: grafiikan peruskomento

Suuri osa tilastollisista peruskuvista voidaan muodostaa komennolla `plot`, jonka yksinkertainen perusmuoto on:

```
> plot(x,y)
> plot(y ~ x)   ## Vaihtoehtoinen muoto
```

Vaaka-akselin arvot annetaan vektorissa `x` ja pystyakselin vektorissa `y`. Mikäli kumpikin muuttuja on numeerinen, havaintopisteet merkitään kuvaan pisteillä koordinaattiensa mukaan.

Komennolle `plot` voi kuitenkin antaa monia lisämääreitä:

- `type`: kuvan tyyppi joka voi olla `p` pisteille (oletus), `l` (pikku-L eikä ykkönen!) viivadiagrammille, `b` sekä viivoille että pisteille, `s` ja `S` askelviivoille sen mukaan onko nousu askelen lopussa vai alussa, `h` "impulssiploteille" ja lopulta `n` jos ei haluta plottausta lainkaan. Viimeisessäkin tapauksessa piirretään akselisto ja niihin voi myöhemmin sijoittaa haluamaansa lisämateriaalia. Viivoja piirrettäessä on huomattava, että pisteet yhdistetään luetellussa järjestyksessä. Normaalissa graafissa on siis pidettävä huolta, että pisteet ovat x–akselilla suuruusjärjestyksessä ja y–akselin pisteet ovat x–akselin mukaisessa järjestyksessä (§3.2)
- `xlab`, `ylab`, `main`, `sub`, `mtext`: Akselien tekstit sekä kuvan pää- ja alaotsikot. Tekstin voi kirjoittaa joko sellaisenaan lainausmerkkien väliin tai komennolla `expression` pystyy lisäämään myös matemaattisia erikoismerkkejä ja -merkintöjä.
- `xlim`, `ylim`: Akselien päätepisteet, jollei tyydytä oletuksiin. Jos haluamme, että y–akseli varmasti alkaa nollasta, voimme kirjoittaa `ylim = c(0,max(y))`.
- `log`: Logaritmisesti piirrettävät akselit, esim `log="xy"`, jos halutaan molemmat akselit logaritmisiksi.
- `bg`, `fg`, `col`: Taustan, etualan ja piirrettävien symbolien värit. Värit voi ilmoittaa joko yksinkertaisina englanninkielisinä sanoina (`"blue"`, `"thistle2"`: täydellinen lista komennolla `colors()`) tai numeroina `1...15` perusväreille tai joillain lukuisista väriskeemoista sävyväreille (katso `help(colors)`, `help(palette)`).
- `pch`: Pisteille käytetyn symbolin numero vakiosymboleille tai yksi kirjain tai numero, jota käytetään tietylle luokalle.
- `lty`, `lwd`: Viivatyyppi ja viivan paksuus. `lty=1` on yhtenäinen viiva (värillä `col`) ja isommat numerot ovat erilaisia katkoviivoja.

4.2 Kuvan täydentäminen uudella aineksella

Komento `plot` piirtää peruskuvan ja akseliston. Tähän kuvaan voi lisätä uusia kuva-aineksia. Peruskomentoja ovat:

- `points(x,y,...)`: Lisää pisteet koordinaatteihin. Kolme pistettä (`...`) tarkoittaa, että lisäksi voi antaa komennon `plot` määreitä kuten `pch`, `col`, ...

- `lines(x,y,...)`: Lisää murtoviivan.
- `matplot(x,mat,...)`, `matlines`, `matpoints`: Lisäävät parametrina annetun matriisin `mat` kaikki sarakkeet vektorin `x` arvoja vastaan.
- `text(x,y,text,...)`: Lisää tekstiä koordinaattipaikkoihin. Jos halutaan pitemmät kuin yhden merkin mittaiset koodit, voidaan kuva piirtää aluksi määritteellä `type="n"` ja lisätä koodit tällä komennolla.
- `abline(h=c)`, `abline(v=c)`, `abline(a,b)`: Ensimmäiset muodot piirtävät vaakasuoran tai pystysuoran viivan koordinaattiin `c`, viimeinen taas piirtää suoran, jonka yhtälö on $a + bx$.
- `legend(x,y, legend, lty, pch, ...)`: Lisää "legendan" eli käytettyjen viivatyyppien (`lty`) ja symbolien (`pch`) selitykset.
- `segments(x0,y0, x1,y1)`, `arrows(x0,y0, x1,y1)`: Komennot piirtävät viivan pisteestä (x_0, y_0) pisteeseen (x_1, y_1). Komento `arrows` piirtää lisäksi viivaan nuolenpään, joka voi olla joko alussa, lopussa tai molemmissa päissä (parametri `code`).

Interaktiivinen komento `locator` käynnistetään

```
> locator()
```

Kun tämän jälkeen hiiren vasemmalla näppäimellä klikkaa kuvaa, kuuluu vain piippaus, ja kun lopulta näpäyttää hiiren oikeaa näppäintä, komento palauttaa klikattujen pisteiden koordinaatit (jotka tietysti voi tallettaa myös muuttujaan). Komennon avulla pystyy helposti sijoittamaan grafiikkaelementtejä haluamiinsa paikkoihin kuvaa. Toinen interaktiivinen komento `identify` taas lisää suoraan informaatiota havaintopisteisiin. Sitä kutsuttaessa on annettava kuvan x– ja y–koordinaatit sekä lisättävä teksti (`label`):

```
> identify(x,y,label, ...)
```

Jos `label` jätetään pois, `identify` lisää järjestysnumeron (katso kuva 7, sivu 43). Jälleen hiiren oikea näppäin lopettaa komennon.

4.3 Muita kuvakomentoja eri tarkoituksiin

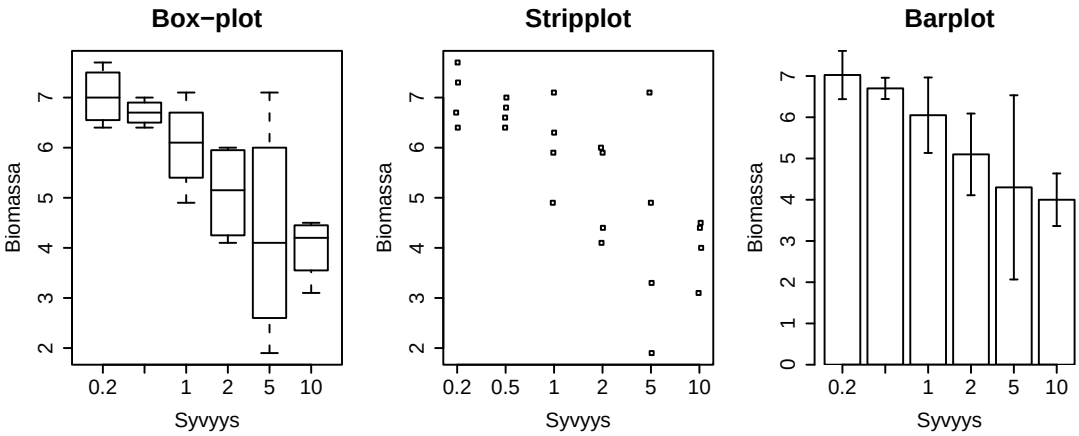
Suurimman osan kuvista voi toteuttaa komennolla `plot`, mutta eräitä kuvatyyppejä varten on vaihtoehtoisia piirroskomentoja tai sopeutettuja versioita `plot`–komennosta. Esittelen vain joitain keskeisimpiä tässä monisteessa käytettyjen diagrammien tyyppejä: on-line dokumentaatio tuntee paljon lisää.

4.3.1 Sirontakuvio kaikkien muuttujien välillä: `pairs`

Komennon `pairs` argumentin on oltava joko aineisto (`data.frame`) tai matriisi — jos nämä ovat komennon `plot` ainoana argumenttina, kutsutaan itse asiassa juuri komentoa `pairs`. Komento piirtää sirontakuviot kaikille muuttujapareille. Liiallinen muuttajien määrä johtaa helposti lukukelvottomaan ja hitaasti syntyvään kuvaan. Kuva on erityisen hyödyllinen tarkasteltaessa muuttujien välisiä suhteissa regressiomalleja rakennettaessa.

Komennon `pairs` kuvapaneeleihin voi määritellä paneelifunktion (`panel`), jonka voi myös määritellä erikseen lävistäjälle tai sen ylä- tai alapuolelle (katso `help(pairs)`). Ilman omaa ohjelmointiakin on tarjolla funktio `panel.smooth`, joka lisää pistejoukkoon tasoittavan käyrän. Tämä on erityisen hyödyllinen kun tutkimme alustavasti muuttujien riippuvuuksia regressioanalyysiä varten. Jos olisimme esim. ensin antaneet komennon:

```
library(rekola) ## Aineistot käyttöön, myös muihin esimerkkeihin
data(plutakko)
pairs(plutakko, panel=panel.smooth)
```



Kuva 1: Sama luokiteltu aineisto (`pojoviken`) esitettynä kolmella vaihtoehtoisella tavalla.

tuskin olisimme aluksikaan kokeilleet suoraviivaista lineaarista regressiota kuten teimme luvussa 7.3 (sivu 39).

4.3.2 Luokitellut selittäjät: `boxplot` ja `stripchart`

Box–plotit ovat kuvia, jotka esittävät arvojen jakauman hyvin selkeästi. Box–plot perustuu niihin viiteen pisteeseen, jotka tulostuvat Tukeyn `fivenum`–komennolla: minimi, noin alakvartiili, mediaani, noin yläkvartiili, maksimi. Kvartiilirajat muodostavat laatikon ("box"), jonka keskellä on vaakaviiva osoittamassa mediaania. Laatikon reunat yhdistetään janoilla maksimiin ja minimiin ja kaikkein äärimmäisimmät havainnot näytetään erikseen pisteinä (kuva 8A, sivu 49). Jos selittävä tekijä on luokkamuuttuja (faktori), komento `plot` tuottaa boxplotin. Komentoa voi kutsua myös julkisesti:

```
> data(pojoviken)
> attach(pojoviken)
> boxplot(biomassa ~ syvyys)  ## tai plot(syvyys, biomassa)
```

Määritteellä `notch=T` komento piirtää vielä "vyötärön", jonka avulla voi arvioida ryhmäkohtaisten mediaanien virhemarginaalien suuruutta.

Box–plot on erittäin hyödyllinen muuttujien jakauman alustavaan tarkasteluun. Sen avulla on helppo huomata mahdollinen vinous sekä poikkeavat havainnot. Yllä olevassa esimerkissäkin näkyy, että eläinplanktonin `biomassa`n hajonta kasvaa kun `syvyys` kasvaa (kuva 1), mikä tekee myöhemmästä varianssianalyysistämme kyseenalaisen (§7.9).

Mikäli aineisto on kovin pieni (kuten melkein kaikki tämän oppaan aineistot), kaikki havainnot esittävä pistekuvio `stripchart` on informatiivisempi (kuva 1):

```
stripchart(biomassa ~ syvyys, vertical=T, method="jitter")
```

Parametri `jitter` siirtelee satunnaisesti pisteitä sivuttain niin että samaan arvoon osuvat havainnot eivät mene pahasti päällekkäin. Vaihtoehtoisesti `method="stack"` sijoittaa samanarvoiset havaintopisteet säännöllisin välien rinnakkain (tai horisontaalisessa kuviossa alekkain).

4.3.3 Pylväsdiagrammi: `barplot`

Komento `barplot(height,...)` piirtää pylväät vektorin `height` arvojen mukaan. Pylväät voivat olla vaaka- tai pystysuorassa ja ne voivat olla hierarkisissa ryhmissä. Komentoa käytetään etenkin

keskiarvojen esittämiseen (kuva 8B). Komento ei kuitenkaan osaa itse laskea keskiarvoja, vaan ne on laskettava syöttötietona annettavaan vektoriin. Tämä tapahtuu esim. komennolla `tapply`.

Pylväisiin on tapana lisätä janat, jotka osoittavat hajonnan tai keskivirheen suuruuden. Nämä janat voi lisätä komennoilla `segments` tai `arrows(..., code=3, angle=90, ...)`. Komennon `arrows` lisämääreet tekevät nuolet janan molempiin päihin ja nuolensakaran kulmaksi nuoleen nähden 90° eli nuolenpää onkin poikkiviiva. Kummatkin komennot vaativat alku- ja loppupisteiden koordinaatit. Komento `barplot` palauttaa pylväiden keskiviivan sijainnin x–koordinaatilla. Esimerkissä tallennamme nämä muuttujaan `mp` ja oletamme symmetrisen virheen suuruuden olevan muuttujassa `jana`:

```
> height <- tapply(biomassa, syvyys, mean) # Keskiarvot
> jana <- tapply(biomassa, syvyys, sd)     # Hajonnat
> mp <- barplot(height, ylim=c(0,max(height+jana)))
> arrows(mp,height-jana, mp,height+jana, code=3, angle=90)
```

Komento `tapply` esitellään tarkemmin luvussa 5.1: pitäkäämme sitä toistaiseksi magiana, jolla saamme piirrettävää. Syntynyttä kuvaa kannattaa verrata samasta aineistosta piirrettyyn box–plottiin (kuva 1). Box–plot on epäparametrinen ja näyttää myös aineiston jakauman vinoudet ja omituisuudet, kun taas perinteinen pylväsdiagrammi pelkistää aineiston kahteen parametriin: keskiarvoon ja hajontaan. Pylväsdiagrammi on adekvaatti, jos aineiston voi adekvaatisti (tai jopa "tyhjentävästi") kuvata näillä kahdella tunnusluvulla.

4.3.4 Histogrammit: `hist` ja `truehist`

Histogrammit ovat ulkoisesti samanlaisia kuin pylväsdiagrammit eli ne ovat pylväitä. Histogrammissa pylvään korkeus kuitenkin edustaa muuttujan arvojen frekvenssiä tai osuutta koko aineistosta. Histogrammi muodostetaan yhdestä vektorista, joka tarpeen vaatiessa jaetaan pätkiin ja havaintojen lukumäärä tai osuus kussakin pätkässä esitetään. Histogrammit ovat muuttujien jakaumien tarkasteluun käytetty työkalu.

Komento `hist` on perus-R:n piirto-ohjelma ja `truehist` on `MASS`–paketin (Venables & Ripley, 1999) työkalu. Kummankin oletuskuva saadaan samanlaisella kutsulla, eli seuraavat komennot piirtävät muuttujan `x` histogrammin:

```
>  x <- rexp(100)  ## Jotain piirrettävää
>  hist(x)
>  library(MASS)   ## truehist
>  truehist(x)
```

Kummassakin komennossa voi kontrolloida pylväiden lukumäärää ja jakoväliä. Kumpikin näistä vaikuttaa suuresti jakauman ulkonäköön (kuva 2), joten kannattaa todellakin kokeilla erilaisia vaihtoehtoja:

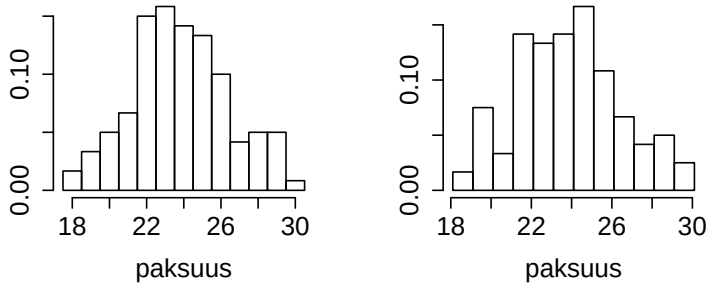
```
>  attach(plutakko)  ## Paketti rekola, kuvaus ■7.3
>  op <- par(mfrow=c(1,2)) ## Kaksi kuvaa rinnan
>  hist(Fotosynt, h=0.5, x0=0) ## jakoväli h, alku x0
>  hist(Fotosynt, h=0.5, x0=-0.2)
>  par(op)  ## Vanhat kuva-asetukset takaisin
```

Venables & Ripley (1999) käsittelevät kuvan parametrointia perusteellisemmin.

4.3.5 Jatkuvan jakauman tiheys: `density`

Komento `density` ei itse asiassa piirrä kuvaa vaan laskee tiedot, joita `plot` tarvitsee piirtääkseen jatkuvan tiheysjakauman:

```
> plot(density(Fotosynt))
```



Kuva 2: Männyn läpimittojen jakauma (Ranta *et al.*, 1989, Esimerkki 2.3, kuva 2.4): Ainoa ero on jakovälien alkukohta.

Tiheyskuva on histogrammin vaihtoehto, jota käytetään ennen kaikkea jatkuville jakaumille. Kuten `hist` ja `truehist`, myös `density` tarvitsee vektorin ja sen perusteella piirtää kuvan arvojen todennäköisyystiheydestä. Jos meillä on kokonaislukumuuttuja (tai muu diskreetti muuttuja), meidän olisi käytettävä epäjatkuville muuttujille tarkoitettuja histogrammikomentoja.[4]

Tiheysjakauma välttää ainakin yhden histogrammin ongelman: jakovälien alkupisteen valinnan. Tiheyskuvissa käytetään liikkuvaa ikkunaa, joten tiheys arvioidaan jatkuvasti eikä pykälittäin. Perinteisistä menetelmistä liukuva keskiarvo on hieman samantapainen, mutta siinä ikkuna on jyrkkärajainen, joten käyrästä ei tule tasainen. Tiheyskuvassa sen sijaan käytetään tasoittavaa ydintä ("kernel"), joka on muodoltaan esim. normaalijakauman tapainen: kun tiettyyn x–pisteeseen arvioidaan tiheyttä, lähimmillä pisteillä on suurin paino ja kauemmilla pienempi. Näin käyrä on tasaisempi. Toista histogrammien ongelmaa ei kuitenkaan pystytä kiertämään: ikkunan leveys ("*bandwidth*") vaikuttaa tasoituksen voimakkuuteen. Jos ikkuna on hyvin kapea, jokainen piste näkyy huippuna käyrässä, jos taas kovin leveä, hävitetään todellistakin informaatiota tiheydestä.

Komento `density` piirtää jakauman siten, että ääripäissään käyrä yhtyy x–akseliin. Jos meillä on aineisto, jossa on runsaasti nollan lähellä olevia arvoja tai jakauma, jonka moodi on nollassa (esim. eskponentiaalinen jakauma), `density` ei esitä ääripäätä oikein.

4.3.6 Jakaumien vertailut: `qqnorm` ja `qqplot`

Komento `qqnorm(x)` piirtää kuvan, jonka x–akselina on normaalijakauman teoreettiset kvantiilit ja y–akselina muuttujan `x` havaitut kvantiilit (kuva 4). Jos muuttuja on likipitäen normaalisti jakautunut, havaintopisteet muodostavat suoran viivan (mikäli suorititte komennon äsken lasketulla muuttujalla `x`, kuvio ei ole suora). Vertailua helpottamaan kuvaan voi lisätä suoran joka kulkee ensimmäisen ja kolmannen kvartiilin kautta käyttämällä komentoa `qqline(x)`.

Komennon `qqplot(x,y)` avulla voi piirtää kaksi jakaumaa toisiaan vastaan. Sitä voi käyttää myös kahden havaitun jakauman vertailuun. Komento `ppoints(n)` jakaa yksikkövälin tasan `n` pisteelle. Tätä funktiota voi käyttää hyväksi tarkasteltaessa havaittua jakaumaa muita kuin normaalijakaumaa vasten. Muuttujamme `x` laskettiin äsken eksponentiaalisesti jakautuneena satunnaismuuttujana (komento `rexp`) ja voimme tarkastella sitä eksponentiaalijakaumaa vasten komennolla:

```
> qqplot(qexp(ppoints(length(x))), x)
```

Missä komento `qexp` palauttaa todennäköisyyspisteitä `ppoints` vastaavat muuttujan arvot.

[4] `plot(table(rpois(100,3.2)), type="h")` piirtää oikeaoppisen kuvan kutsussa lasketusta Poisson-muuttujasta.

4.3.7 Kuvat luokittain: `coplot` ja `lattice`

Komento `coplot` esittää monta erillistä sirontakuviota jonkin luokkatekijän jakojen mukaan tai jatkuvan tekijän ositettuna (esim. kuvat 9, 10). Kaikkien kuvien asteikot ovat samat, joten eri kuvien suora vertailu on helppoa. Komennon yleinen muoto on

```
> coplot(y ~ x | luokka)
```

missä `luokka` on luokitteleva muuttuja. Myös komennossa `coplot` voi olla parametri `panel`, jolle on tarjolla funktio `panel.smooth`. Käsittelemme myöhemmin oman paneelifunktion kirjoittamista (§7.9, §7.10).

Vaihtoehtona `coplot`-komennolle voi käyttää kirjaston `lattice` komentoja, jotka ovat samantapaisia kuin S-PLUS -ohjelman `trellis`-funktiot. `Lattice`-komennot ovat erittäin monipuoliset ja voimakkaat. Niitä kannattaa opiskella kirjaston omilta apusivuilta.

4.4 Yleinen parametrointi ja ympäristö

Kuvien parametreja voi muuttaa monella tapaa. Osa peruskomennon `plot` yhteydessä esittämistäni parametreista (§4.1) on itse asiassa yleisiä kuvaparametreja, jotka voi antaa myös erikseen komennolla `par`. Selvittääkseen kuvakomennon mahdollisuudet, ei yleensä riitäkään että katsoo komennon ohjesivuja, vaan on myös katsottava

```
> help(par)
```

Olemme jo usein kohdanneet yhden `par`–käskyn, jota tarvitsemme usein:

```
> op <- par(mfrow=c(2,2)) # 2 x 2 osakuvaa, riveittäin
> par(mfcol=c(2,3))       # 2 x 3 osakuvaa, sarakkeittain
> par(op)                 # Palautetetaan vanhat parametrit
```

Parametrit `mfrow` ja `mfcol` määräävät piirrettäväksi monta osakuvaa samalle sivulle tai näytölle. Komento `par` palauttaa vanhat arvonsa, ja jos nämä talletetaan, voidaan muutosten jälkeen palata alkuperäisiin asetuksiin antamalla talletetut arvot parametrina. Parametreja on todella paljon; niihin on paras tutustua R:n ohjesivuilta.

Osa parametreista koskee vain joitain tulostuslaitteita. Esimerkiksi Postscript–kuvien muotoon vaikuttavat parametrit `ps.options`. Myös joillain yleisillä optioilla (`options`) voi olla vaikutusta kuviinkin.

Kuvat piirretään yleensä ensin näytölle erilliseen grafiikkaikkunaan. Kun kuva on valmis, sen voi kopioida muille laitteille eli “deviiseille”. Komento `dev.copy` kopioi kuvan asetuksissa säädetelly tai komennossa ilmoitetulle “laitteelle”, joka voi olla myös tiedosto. Joillekin suosituille vaihtoehdoille on omat komentonsa. Esim. tämä oppaan kuvat ovat EPS–tiedostoja, joita varten on oma komento:

```
> dev.copy2eps(width=5, height=4) # Kuvan koko tuumina
```

Komento `dev.print` (yrittää) tulostaa kuvan kirjoittimelle — sen on tietysti oltava Postscript–kykyinen ja R:n on oltava määritelty yrittämään tulostusta. Muussa tapauksessa kuva yleensä tulostetaan Postscript–tiedostoksi.

5 Aineiston alustava tarkastelu

Uutta aineistoa käsiteltäessä, sitä on aina syytä tarkastella ennen kuin ryhtyy varsinaisiin analyyseihin. Liiallisuuksiin tässä ei toki pidä mennä. On kuitenkin hyvä saada jonkinlainen kuva aineistosta ennen kuin ryhtyy sitä käsittelemään: muuttujien jakaumista, riippuvuussuhteista jne.

Esimerkkeinä tässä luvussa käytämme kahta myöhemmin esiteltävää aineistoa paketista `rekola`: `plutakko` (§7.3), jossa on biologisia ja kemiallisia mittauksia kalliolammikoista, sekä `taimet` (§7.4) missä on eri tavoin istutettujen mäntyjen pituuksia.

Meillä on kaksi käyttökelpoista komentoa, jotka kertovat meille paljon aineistosta: `summary` ja `plot`.

```
>  data(plutakko)
>  summary(plutakko)
>  plot(plutakko)
```

Komento `summary` tulostaa kustakin muuttujasta viisi peruskvanttiilia: minimin, alakvartiilin, mediaanin, yläkvartiili ja maksimin sekä lisäksi vielä keskiarvon. Näiden lukujen perusteella näkee muuttujien vaihtelualueen ja voi saada jonkinlaisen kuvan lukujen jakaumasta. Komento `plot` taas tässä tapauksessa piirtää sirontakuviot muuttujista kaikkia muita vastaan, joten saamme jonkinlaisen kuvan riippuvuussuhteista. Näiden kahden komennon avulla pääsee jo pitkälle — joskus muuta ei kaivatakaan. Lisäksi R:ssä on kyllä koko joukko muitakin mahdollisuuksia.

5.1 Tunnusluvut

R:ssä on suuri joukko komentoja erilaisten tilastollisten tunnuslukujen laskemiseksi. Joitain tärkeimpiä ovat: `mean` (keskiarvo), `var` (varianssi), `sd` (hajonta), `median` (mediaani), `quantile` (kvantiileja), `fivenum` (Tukey viisi tunnuslukua, melko lähellä kvanttiileja) jne. Nämä tunnusluvut lasketaan vektoreille. Mikäli haluamme saada matriisin tai aineiston riveille tai sarakkeille nämä vektoritunnusluvut, meidän on sovellettava komentoa `apply`, jonka muoto on

```
> apply(MATRIISI, REUNA, FUNKTIO)
```

Muut termit tässä ovatkin selviä, mutta `REUNA` vaatinee selvityksen: R:ssä kaksiulotteinen matriisi esitetään `x[i,j]`, missä i on ensimmäinen `REUNA` eli rivit ja `j` on toinen `REUNA` eli sarakkeet. Jos siis haluamme soveltaa jotain funktiotamme sarakkeille, kirjoitamme

```
> apply(plutakko, 2, quantile)
     Fotosynt Pikopl Plankt   NH4     PO4
0%    0.14700 0.0000 0.1400 2.000 0.43800
25%   0.35375 0.2725 0.2525 3.150 0.65700
50%   1.18050 0.5650 0.5800 4.450 1.25950
75%   2.10700 1.7625 1.9600 5.425 1.59825
100%  4.68200 2.9900 6.1600 8.900 4.50000
```

Mikäli meillä on aineistossa luokkamuuttujia, tarvitsemme usein tunnusluvut luokittain. Silloin voimme käyttää edellisen kaltaista funktiota `tapply`, jonka muoto on

```
> tapply(MUUTTUJA, LUOKITTELIJA, FUNKTIO)
```

Tällä kertaa meidän on kutsuttava funktiota yhdelle vektorille (muuttujalle) kerrallaan, ja tämä luokitellaan samanmittaisen vektorin `LUOKITTELIJA` perusteella:

```
> data(taimet)
> attach(taimet)
> tapply(pituus, istutus, median)
 kenno ruukku  rulla
    49     55     51
```

Komennolla `table` saadaan luokka- tai kokonaislukumuuttujan frekvenssitaulukko:

```
> table(pituus) # Männyn tainten pituudet
pituus
46 47 49 50 51 52 53 54 55 58 61
 1  1  2  2  2  1  1  2  1  1  1
> table(rpois(100, 0.9)) # 100 Poisson(mu=0.9) satunnaislukua
 0  1  2  3  4
45 36 10  6  3
```

Jatkuvan muuttujan pystyy jakamaan paloiksi komennolla `cut`.

R:ssä on kiinnitetty melko paljon huomiota robusteihin menetelmiin, eli sellaisiin, jotka kestävä aineiston poikkeamisen oletuksista. Niinpä `summary` tulostaa ennen kaikkea kvantiileja eikä keskihajontaa, vinoutta ja kurtoosia, kuten usein on tapana. Myös monista perustunnusluvuista on robusteja variantteja. Esim. keskiarvot (`mean`) voivat olla trimmattuja, eli osa äärimmäisistä havainnoista poistetaan. Venables & Ripley (1999) käsittelevät robustia estimointia perusteellisemmin ja paketissa `MASS` on lisää robusteja tunnuslukuja.

5.2 Jakaumien tarkastelu

R:ssä on suuri joukko graafisia komentoja jakaumien tarkastelemiseksi (§4.1): erityisen hyödyllisiä ovat komennot `boxplot`, `pairs`, `hist`, `truehist`, `qqnorm` ja `qqplot`. Näitä komentoja kannattaa käyttää saamaan jonkinlainen kuva muuttujien jakaumista. Jakaumaoletukset koskevat kuitenkin jäännösvaihtelua mallin suhteen, joten lopulliset jakaumakorjaukset on tehtävä mallinsovituksen jälkeen (§7.13). Alustava tarkastelu antaa kuitenkin aavistuksen myös sovituksenjälkeisistä jakaumista.

R:ssä jakaumien tarkastelu pohjatuu ennen kaikkea graafisiin arviointeihin. Joitain testejä voi toki käyttää (§6.2), mutta nämä ovat epäherkkiä pienillä aineistoilla ja turhankin herkkiä suurilla aineistolla: kaikki havaitut jakaumat poikkeavat odotetusta jos vain aineisto on kyllin iso. R:ssä *ei* käytetä vinouden ja huipukkuuden tunnuslukuja kuten monessa muussa ohjelmassa. Keskiarvoa voidaan pitää aineiston ensimmäisenä momenttina (perustuu x–termeihin), varianssia ja keskihajontaa toisena momenttina (perustana x^2–termit), vinoutta taas kolmantena ja huipukkuutta ("kurtositeetti") neljäntenä momenttina. Graafiset tarkastelut ovat selkeämpiä kuin nämä kolmannet ja neljännet momentit.[5]

Jakaumien tarkastelemiseen voi graafisten komentojen sijaan tai lisäksi käyttää myös tekstipohjaista komentoa `stem`, joka tulostaa ns. runko-lehtikuvion ("*stem and leaves plot*"):

```
> attach(taimet)
> stem(pituus) # Tainten pituus aineistossa taimet

  The decimal point is 1 digit(s) to the right of the |

  4 | 6799
  5 | 00112344
  5 | 58
  6 | 1
```

Tästä voimme lukea, että tainten pituudet olivat $46, 47, 49, 49, 50, \ldots, 61$ cm eli näemme sekä (melko) tarkat luvut että karkean "histogrammin" jakaumasta.

[5] Tukeyn "*power of five*" –sääntö sanoo: "Älä kuvittelekaan estimoivasi k:nnetta momenttia ellei aineistosi koko ole vähintään 5^k havaintoa" eli keskiarvolle 5, hajonnalle 25, vinoudelle 125, huipukkuudelle 625,...

6 Perinteiset testit ja luottamusvälit

Perinteiset testit täyttävät monet biometrian oppikirjat, mutta kuuluvat jollain tapaa tilastotieteen periferiaan: monet ovat irrallisia temppuja, jotka eivät liity kovinkaan likeisesti toisiinsa saati muihin tilastollisiin rakennelmiin. Ilmeisesti tämän takia monet näistä testeistä on R:ssä sisällytetty erilliseen pakettiin `ctest` ("*classical tests*"), joka kuitenkin on automaattisesti aina käytössä. Esittelen näistä perinteisistä testeistä vain t-testin sekä joitain jakaumien yhteensopivuustestejä. Monet Biometrian kirjan (Ranta *et al.*, 1989) esittelemistä testeistä ovat toki saatavillla, ja lisäksi koko joukko muita. Ajantasaisen listan käytettävistä testeistä näkee komennolla

```
> library(help=ctest)
```

6.1 t-testi

Perinteisistä testeistä keskiarvojen vertailuun käytetty t-testi on kaikkein suosituimpia. t-tunnusluvun testaaminen perustuu sinänsä normaalijakaumaan ja t-testillä on sijansa myös lineaaristen mallien kertoimien merkitsevyyden arvioimisessa (§7). Vaikka perinteinen t-testi onkin paketissa `ctest`, t–tunnusluvun arviointi on toki perus-R:ssä (funktiot `pt` ja `qt`). Sen sijaan testisuureiden johtaminen on perinteisen testin arvoinen.

Jotta turha mystiikka karisisi t-testin ympäriltä, esitän sen periaatteen yksinkertaistetussa muodossa:

$$t_{df} = \frac{\text{ero}}{\text{keskivirhe}}$$

t-testissä verrataan keskiarvon ja jonkin vakioarvon tai kahden keskiarvon *eroa*. Tämän eron suuruutta arvioidaan *keskivirhettä* vasten. Jos keskiarvon jakauma on normaalinen, kuten keskiarvon jakauma aina asymptoottisesti on, satunnaisvirheen keskimääräisen suuruuden voi arvioida keskiarvon keskivirheenä ja tällöin t-jakauma vapausasteilla df antaa todennäköisyyden saada sattumalta havaitun suuruinen tai suurempi ero. Mikäli tämä todennäköisyys on pieni, hylkäämme nollahypoteesin '*todellinen ero = 0*' eli sanomme eron olevan "merkitsevän". "Merkitsevyyden" testaaminen on ekologiassa liiankin suosittua. Usein on paljon hyödyllisempää tarkastella estimaattien luottamusvälejä, jotka antavat paremman kuvan otoskeskiarvon luotettavuudesta.

t-testin lukuisat variantit eroavat toisistaan siinä kuinka *keskivirhe* on arvioitu:

- *Otoskeskiarvoa $\bar{x}$ verrataan populaatiokeskiarvoa eli odotusarvoa μ koskevan nollahypoteesin mukaiseen lukuun μ_0* (jolloin ero $= \bar{x} - \mu_0$): Jakajana on keskiarvon keskivirhe ja vapausasteita on $n - 1$ eli keskiarvon laskemiseen käytettyjen havaintojen luku (n) miinus estimoitujen keskiarvojen lukumäärä (1).

- *Kahta otoskeskiarvoa verrataan keskenään*: Jakajana on keskiarvojen erotuksen keskivirhe. Tällöin joudutaan menettelemään eri tavoin sen mukaan onko ns. yhdistetyn (*pooled*) otosvarianssin (tai otoskeskihajonnan) laskeminen oikeutettua vai ei:

  - *Teoreettiset varianssit σ_1^2 ja σ_2^2 taustalla olevissa populaatioissa ovat samansuuruiset*: lasketaan keskivirhe yhdistetystä otosvarianssista. Vapausasteet ovat $n_1 + n_2 - 2$ eli havaintojen yhteismäärä ($n_1 + n_2$) miinus estimoitujen keskiarvojen määrä (2).

  - *Teoreettiset varianssit populaatioissa ovat eri suuret*: yhdistettyä otosvarianssia ei voi laskea. Tällöin lasketaan keskivirhe erillisistä otosvariansseista ja synnin hyvitykseksi vähennetään vapausasteiden määrää, sitä enemmän mitä erilaisemmat otosvarianssit ja ryhmien koot (Welchin menetelmä).

Kaikki nämä (ja eräät muutkin) tapaukset voidaan käsitellä komennolla `t.test`.

6.1.1 Keskiarvon vertaaminen teoreettiseen arvoon

Keskiarvon vertaaminen teoreettiseen arvoon eli populaatiokeskiarvoa μ koskevaan nollahypoteesiin μ_0 on kaikkein yksinkertaisin tapaus, sillä silloin virheemme on yksiselitteisesti ainokaisen keskiarvon keskivirhe. Sen sijaan ekologiassa ei kovin usein tule vastaan perustestitilanteita, missä teoreettinen arvo on selkeästi määritelty. Sen sijaan lineaaristen mallien (§7) parametreja testataan usein teoreettista arvoa 0 vastaan.

Biometrian kirjan esimerkki (Ranta *et al.*, 1989, Esimerkki 7.2) käsittelee varpusen nilkan pituutta (mm) Herttoniemessä ja Helsingin keskustassa. Vähäinen aineisto on helppo kirjoittaa suoraan R-istunnossa:

```
> city <- c(21.4, 15.9, 20.3, 19.7, 17.0, 16.4)
> hertto <- c(18.1, 16.5, 22.2, 14.7, 13.9, 16.7, 13.5)
```

En tunne mitään erityistä kunnianhimoa kehittää teoriaa varpusen nilkan pituudesta, mutta kuvitelkaamme, että tutkija miettii, voiko hän puhtain tunnoin sanoa varpusen nilkan pituuden odotusarvon ao. populaatiossa olevan 16 mm pitkä:

```
> varpunen <- c(city,hertto)  ## varpuset, liittykää yhteen!
> t.test(varpunen, mu=16)

        One Sample t-test

data:  varpunen
t = 1.8262, df = 12, p-value = 0.09278
alternative hypothesis: true mean is not equal to 16
95 percent confidence interval:
 15.72822 19.08716
sample estimates:
mean of x
 17.40769
```

Tutkitun 13 varpusen nilkanpituus on keskimäärin 17.4 mm mikä ei näemmä merkitsevästi poikkea "teoriastamme" 16 mm nilkasta. Itse asiassa `t.test` tulostaa myös havaitun keskiarvon 95 % luottamusvälit 15.8 . . . 19.1 mm. "Teoreettinen" arvomme sisältyy tähän luottamusväliin, joten se voisi olla mahdollinen tosi arvo, kuten myös esim. 19 mm. "Merkitsevyystestin" sijaan hyödyllisempi tieto oli nilkanpituuden luottamusväli. Aineiston luottamusväli on tosin niin väljä, ettemme voi sanoa mitään kovin luotettavaa nilkanpituudesta.

6.1.2 Kahden keskiarvon vertaus

Kahta keskiarvoa verrattaessa meidän on aluksi selvitettävä, voidaanko keskiarvojen erotuksen keskivirhe laskea yhdistetystä otosvarianssista eli ovatko populaatiovarianssit yhtä suuret. Yleensä meillä riittää että varianssit eivät ole kovin erisuuret. Varianssien yhtäsuurutta voi arvoida `ctest` ohjelmalla `var.test`:

```
> var.test(hertto,city)  ## Testi: varianssien suhde = 1?

        F test to compare two variances

data:  hertto and city
F = 1.7012, num df = 6, denom df = 5, p-value = 0.5764
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
  0.243808 10.186162
sample estimates:
ratio of variances
```

```
          1.701219
```

Varianssisuhteen luottamusväli oli hyvin väljä ($0.24 \ldots 10.19$), joten emme voi sanoa paljonkaan populaatiovarianssien suhteista. Toisaalta erillisten tai yhtäläisten varianssien käyttäminen ei paljonkaan vaikuta testisuureisiin, elleivät varianssit ole hyvin erilaiset. Jos taas varianssit poikkeavat voimakkaasti toisistaan, kannattaa pysähtyä miettimään, onko pelkkien keskiarvojen vertailu mielekästä ongelmamme kannalta.

```
> t.test(hertto,city, var.equal=T)  ## Oletetaan yhtäsuuret varianssit

          Two Sample t-test

data:  hertto and city
t = -1.2855, df = 11, p-value = 0.2250
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -5.250008  1.378580
sample estimates:
mean of x mean of y
 16.51429  18.45000
```

Herttoniemen ja keskustan varpusten nilkan pituudet 16.5 mm ja 18.4 mm eivät poikkea merkitsevästi toisistaan ($p = 0.225$). On kuitenkin valaisevampaa tarkastella nilkan pituuksien eron luottamusväliä $-5.3 \ldots 1.4$ mm, jonka mukaan seuraavat tilanteet ovat mahdollisia:

1. Herttoniemen varpusten nilkat ovat keskimäärin lyhyempiä kuin cityvarpusten.
2. Herttoniemen varpusten nilkat ovat keskimäärin yhtä pitkät kuin cityvarpusten.
3. Herttoniemen varpusten nilkat ovat keskimäärin pitempiä kuin cityvarpusten.

Luottamusväleihin sisältyy nollahypoteesi (vaihtoehto 2) joten emme voi hylätä nollahypoteesia nilkkojen yhtäläisyydestä. Toisaalta emme myöskään voi väittää että nollahypoteesi on oikea: se on vain *mahdollinen*, mutta mahdollista on myös että herttoniemeläiset ovat lyhyempi- tai pitempinilkkaisia. Tämän takia ei myöskään ole sallittua sanoa "cityvarpuset olivat pitempinilkkaisia, vaikka ero ei ollutkaan merkitsevä". "Epämerkitsevä ero" tarkoittaa juuri, että kaikki vaihtoehdot ovat mahdollisia emmekä pikkuruisen aineistomme takia pysty sanomaan, minkä suuntainen ero on.

Voimme olla melkoisen varmoja, että eri biologisista populaatioista mitatut muuttujat ovat erisuuria, mutta luontainen vaihtelu on niin suurta, ettemme pysty aineistomme perusteella sanomaan, minkäsuuntainen ero on. Tällaisen biologisen havaintoaineiston analyysi vastaa vain kysymykseen, onko aineistomme riittävän suuri luontaisen eron havaitsemiseen. Hyvin suuressa aineistossa pystymme havaitsemaan vähäpätöisiäkin eroja keskiarvoissa. Sen jälkeen voimme ryhtyä miettimään, ovatko nämä "merkitsevät" erot myös biologisesti *merkittäviä* — mitä ne eivät välttämättä ole. Sen sijaan "epämerkitsevien" erojen pohjalta emme voi edes spekuloida. Jos luottamusvälit ovat ahtaat, voimme sen sijaan argumentoida jopa nollahypoteesin puolesta, jos se sisältyy väliin. Luottamusväli on tulkinnallisesti paljon arvokkaampi tunnuslukupari kuin p–arvo.

Varpusaineistossa saatoimme käyttää yhtäläisten varianssien oletusta. Erisuuret varianssit on oletusvaihtoehtona komennossa `t.test`:

```
> t.test(hertto,city)  ## Sallitaan erisuuret varianssit;  t.testin oletusarvo

         Welch Two Sample t-test

data:  hertto and city
t = -1.3137, df = 10.9, p-value = 0.2159
alternative hypothesis: true difference in means is not equal to 0
```

```
95 percent confidence interval:
 -5.182348  1.310920
sample estimates:
mean of x mean of y
 16.51429  18.45000
```

Koska varianssit olivat melko yhtälaiset, erot testisuureissa ovat pieniä.

6.1.3 Parittainen vertailu

t-testillä voi verrata myös samoista yksiköistä tehtyjä toistuvia havaintoja. Tällainen testi on kuitenkin vain erikoistapaus yhden otoksen *t*-testistä (§6.1.1): tutkittava muuttuja on parittaisten havaintojen *erotus*, jonka keskiarvoa verrataan nollahypoteesiin ero = 0.

Biometrian oppikirjassa (Ranta *et al.*, 1989, Esimerkki 7.10) tutkitaan sekundaariyhdisteiden määrää tunturikoivun lehdissä ennen ja jälkeen tunturimittarihyökkäyksen. Pieni aineisto on jälleen helppo kirjoittaa suoraan istunnossa:

```
> dagen <- c(51.7,54.2,53.3,57,56.4,61.5,57.2,56.2,58.4,55.8)
> efter <- c(62.5,65.2,67.6,69.9,69.4,70.1,67.8,67,68.5,62.4)
```

Näiden salaperäisten lukujen yksikköä tai sekundaariyhdisteiden laatua ei ole erikseen mainittu. Käsitelkäämme niitä siis vain lukuina.

Parittainen *t*-testi tehdään niin ikään komennolla `t.test`:

```
> t.test(efter,dagen, paired=T)  ## Vertailu pareittain

         Paired t-test

data:  efter and dagen
t = 15.4584, df = 9, p-value = 8.683e-08
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  9.279307 12.460693
sample estimates:
mean of the differences
                  10.87
```

Testattava ero on sekundaariyhdisteiden määrän muutos 10.9. Muutoksen luottamusväli on suppea (9.3 . . . 12.5), mikä ei sisällä arvoa 0 eli muutos on merkitsevä. Meillä on vain yksi arvo (muutos), jonka keskiarvo on laskettu kymmenestä havainnosta, joten vapausasteita on 9.

Koska havaintomme olivat pareittain, parittainen *t*-testi kuului tehdä. Tämän pakon lisäksi parittainen testi on usein tehokkaampi: alkuperäisten lukujen varianssi on usein suuri, joten pieni ero jää huomaamatta. Eron varianssi voi silloinkin olla pieni ja tämän parittainen *t*-testi huomaa. Tässä tapauksessa merkitsevyyksien ero tavalliseen *t*-testiin ei ole järisyttävä, mutta parittaisessa testissä saatiin huomattavasti suppeammat luottamusvälit erolle. Koska `var.test` ei pidä mittauskertojen variansseja erisuurina, käytämme yhtäsuurten varianssien oletusta:

```
> t.test(efter,dagen, var.equal=T)  ## 2 keskiarvon vertaus

          Two Sample t-test

data:  efter and dagen
t = 8.744, df = 18, p-value = 6.753e-08
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  8.258267 13.481733
```

```
sample estimates:
mean of x mean of y
    67.04     56.17
```

Esimerkki ei tietenkään osoita, että tunturimittarien hyökkäys olisi indusoinut sekundaariyhdisteiden tuotoksen. Se osoittaa vain, että todennäköisesti kolme päivää myöhemmin sekundaariyhdisteitä on enemmän kuin mittaushetkellä, mutta tämä muutos olisi saattanut tapahtua ilman tunturimittariakin.

6.1.4 Monta yhtaikaista vertailua

Mikäli tehdään monta t-testiä yhtaikaa, on hyvin todennäköistä, että jokin niistä todetaan "merkitseväksi" sattumalta. Käyttäessämme kriittistä rajaa $p = 0.05$ otamme tietoisesti 5 % riskin että saamme erheellisesti merkitsevän tuloksen vaikkeivät keskiarvot eroakaan toisistaan. Jos teemme monta testiä, erheellisen päättelyn todennäköisyys jossain testissä kasvaa. Jos vertaamme kuutta keskiarvoa kaikkia keskenään, teemme $6 \times 5/2 = 15$ vertailua ja tällöin todennäköisyys saada sattumalta ainakin yksi merkitsevä tulos on $1 - (1 - 0.05)^{15} = 0.54$ eli yli puolet.

Monta keskiarvoa yhtaikaa verrattaessa meidän on sopeutettava vertailukriteerimme testien määrään. Kaikkein yksinkertaisin on *Bonferronin menetelmä*, missä p-arvot jaetaan testien määrällä. Tämä yksinkertainen korjaus toimii yllättävän hyvin: esimerkissämme $1-(1-0.05/15)^{15} = 0.049$.

Monen keskiarvon parittaisen vertailun voi tehdä komennolla `pairwise.t.test`. Paketissa `rekola` on pieni aineisto `taimet`, jossa tutkitaan eri istutustapojen (luokkamuuttuja `istutus`) vaikutusta männyntaimien pituuteen (muuttuja `pituus`; aineiston tarkempi kuvaus §7.4). Keskiarvot ovat:

```
> data(taimet)
> attach(taimet)
> tapply(pituus,istutus,mean)
 kenno ruukku  rulla
    49     56     51
```

Kaikkien keskiarvojen yhtaikainen vertailu Bonferronin korjauksella antaa merkitsevyydet:

```
> pairwise.t.test(pituus,istutus, p.adjust.method="bonf")

        Pairwise comparisons using t tests with pooled SD

data:  pituus and istutus

       kenno  ruukku
ruukku 0.0057 -
rulla  0.8418 0.0460

P value adjustment method: bonferroni
```

Toisin sanoen kenno- ja ruukkutaimet erosivat toisistaan, samaten ruukku- ja rullataimet, mutta rullataimet eivät eronneet kennotaimista. Ilman Bonferronin korjausta viimeinen ero olisi:

```
> t.test(pituus[istutus=="kenno"], pituus[istutus=="rulla"], var.equal=T)

          Two Sample t-test

data:  pituus[istutus == "kenno"] and pituus[istutus == "rulla"]
t = -1.3484, df = 8, p-value = 0.2145
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -5.420357  1.420357
```

```
sample estimates:
mean of x mean of y
       49        51
```

Tämä ei myöskään ole merkitsevä, mutta p–arvo on toki paljon alempi. Meillä oli vain kolme yhtaikaista vertailua, joten korjauksen vaikutus ei ollut kovin voimakas. Jos tällaisten korjausten tielle lähtee, on tietysti hyvin vaikea sanoa, milloin lopettaa: pitäisikö jakajana olla yhden vertailun, koko tutkimusprojektin vai uran aikaisten testien määrä? Usein korjaus tehdään liiankin herkästi. Identifioitaessa merkitsevän, moniluokkaisen varianssianalyysin (§7.4) toisistaan poikkeavia luokkakeskiarvoja, korjauksen käyttö saattaa olla oikeutettu ja ainakin se on ekologiassa tavanmukainen.

Tekemämme testi perustui yhdistettyyn keskihajontaan ja oli oikeutettu mikäli variansseja saattoi pitää yhtäsuurina. Varianssien yhtäsuuruuden samanaikaiseen vertaamiseen voi käyttää *Bartlettin testiä*, mutta on muistettava että jälleen testaamme pikemminkin aineiston kokoa. Varianssien suora ja epämuodollinen tarkastelu saattaa olla informatiivisempi kuin testi:

```
> bartlett.test(pituus,istutus)

        Bartlett test for homogeneity of variances

data:  pituus and istutus
Bartlett's K-square = 1.3744, df = 2, p-value = 0.503
```

6.2 Yhteensopivuustestit

Yhteensopivuustesteillä arvioidaan vastaako havaittu jakauma jonkin yksinkertaisen nollahypoteesin perusteella odotettua jakaumaa, esimerkiksi tasajakaumaa tai jotain odotettua teoreettista jakaumaa. Yhteensopivuustesteistä esittelen khi–neliön testin (tai χ^2–testin, jos halutaan käyttää kreikkalaisia kirjaimia) sekä Kolmogorovin–Smirnovin testin.

6.2.1 Khi–neliön testi

χ^2–jakauma on normitettuun normaalijakaumaan pohjautuva jakauma: Jos muuttuja z_i jakautuu normaalisesti siten että keskiarvo on 0 ja varianssi on 1 eli $z_i \sim N(0,1)$, niin n riippumattoman z_i-luvun neliöiden summa jakautuu khi–neliöjakauman mukaan vapausasteilla n:

$$\sum_{i=1}^{n} z_i^2 \sim \chi^2(n)$$

Normaalisesti $N(\mu,\sigma^2)$ jakautuneen muuttujan pystyy muuntamaan $N(0,1)$ jakautuneeksi z-muunnoksella (yhtälö 1 sivulla 8). Jos meillä on Poisson-jakautunut muuttuja $y \sim P(\mu)$ niin sekä sen varianssi että keskiarvo ovat μ. Mikäli odotusarvo μ on kyllin suuri, voimme aproksimoida Poisson-jakaumaa normaalijakaumalla ja silloin normeeraava z–muunnos on:

$$z = \frac{y-\mu}{\sqrt{\mu}} \tag{3}$$

Lukumäärien oletetaan yksinkertaisissa malleissa tyypillisesti olevan Poisson-jakautuneita, joten voimme käyttää tämän kaavan lauseketta. Silloin n:n riippumattoman z-muunnoksen neliöiden summa on jakautunut likimain $\sim \chi^2(n)$-jakauman mukaan. Näin ollen voimme käyttää yhtälöä 3 testaamaan havaittujen (y) ja odotettujen (μ) lukumäärien eroja. Aproksimaatiota pidetään yleisesti riittävän kun kaikki $\mu > 5$. Kaavat ovat aivan samat kuin Biometrian oppikirjassa (Ranta *et al.*, 1989) vaikka ne johdettiin eri tavoin. Valitsin tämän tavan painottaakseni, että

- χ^2–jakauma ei ole erityisen "epäparametrinen" vaan pohjautuu tiukasti normaalijakaumaan.

- χ^2–jakauma on likeisesti kytkeytynyt myös lineaaristen mallien testaamiseen käytettyyn F–jakaumaan, joka on vapausasteilla skaalattujen χ^2–muuttujien suhde.

Biologit ovat jo pitkään noudattaneet käytäntöä jättää kreikkalainen merkintä χ^2 teoreettisella jakaumalle ja käyttää latinalaista merkintää X^2 sitä likimäärin noudattavasta testisuureesta (Ranta *et al.*, 1989). Ranta *et al.* (1989) kuvaavat myös G^2–testin, joka on Poisson-jakauman uskottavuusosamäärä. Joillain ekologian aloilla sen käyttö on perinne, mutta useimmiten sen hyötyä pidetään marginaalisena enkä minäkään sitä käsittele.

Khi–neliön yhteensopivuustestillä verrataan havaittua jakaumaa odotettuun jakaumaan. Mikäli erityistä syytä ei ole, yksiulotteisessa taulukossa oletetaan odotusjakaumaksi tasajakauma. Oppikirjan pikkuruinen esimerkki (Ranta *et al.*, 1989, Esimerkki 6.1) on mahdollista kirjoittaa suoraan komentoon:

```
> chisq.test(c(39,53,152))

        Chi-square test for given probabilities

data:  c(39, 53, 152)
X-squared = 93.3033, df = 2, p-value = < 2.2e-16
```

eli suutari ei tosiaankaan syö yhtä paljon erikokoisia saaliseläimiä vaan suosii suuria vesikirppuja.

Kaksiulotteisissa taulukoissa testataan yleensä heterogeenisuutta: ovatko rivit keskenään samanlaisia ja ovatko sarakkeet keskenään samanlaisia. Ranta *et al.* (1989, Esimerkki 6.3) käsittelevät tapausta, missä kuudelle eri vesiliskoille tarjotaan ravinnoksi surviaistoukkia tai vesikirppuja ja katsotaan niiden ravinnonvalintaa. Aineisto voidaan kirjoittaa R-istunnossa:

```
> ruoka <- c(17,5, 14,4, 13,4, 3,11, 5,15, 4,12)
> dim(ruoka) <- c(2,6) ## ruoka-vektori matriisiksi
> ruoka
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   17   14   13    3    5    4
[2,]    5    4    4   11   15   12
```

Tavanomainen tapa on kontingessitaulukon analyysi (Ranta *et al.*, 1989), jonka `chisq.test` tekee automaattisesti kun sille annetaan syöttötietona matriisi. Kontingenssitaulukon analyysissä testataan hypoteesia, ovatko kaikkien rivien arvot jakautuneet samoin ja ovatko kaikkien sarakkeiden arvot jakautuneet samoin. Vektoria testattaessa nollahypoteesina (ellei toisin sanottu) oli lukumäärien tasajakauma. Kontingenssitaulukossa sen sijaan jakaumaoletus saadaan *reunasummista*. Esimerkissämme kunkin vesiliskoyksilön ravintopreferenssien oletettiin olevan:

```
> apply(ruoka,1,sum)/sum(ruoka)
[1] 0.5233645 0.4766355
```

Odotusarvot saadaan jakamalla vesiliskon syömä ruoan määrä näillä frekvensseillä. Ensimmäinen vesilisko söi kaikkiaan 22 ravintoeläintä. Jos sen preferenssit olisivat samat kuin kaikkien yksilöiden summattuna, ensimmäinen yksilö söisi $0.523 \times 22 = 11.51$ surviaista (havaittu 17) ja $0.477 \times 22 = 10.49$ vesikirppua (havaittu 5). Mikäli havaitut rivi- ja sarakeprofiilit poikkeavat reunasummien perusteella arvioiduista , sanotaan aineiston olevan heterogeeninen:

```
> chisq.test(ruoka)

        Pearson's Chi-square test

data:  ruoka
X-squared = 30.2696, df = 5, p-value = 1.305e-05
```

Saatamme hylätä suurella varmuudella nollahypoteesin. Vesiliskoyksilöiden ravintopreferenssit poikkeavat toisistaan suurella varmuudella.

Ranta *et al.* (1989) esittävät tämän yksinkertaisen analyysin sijaan sangen mutkallisen heterogeenisuustestin. Mielestäni ylläoleva suoraviivainen kontingenssitaulukon analyysi antaa paremman vastauksen selkeämpään kysymykseen kuin kirjan heterogeenisyystesti. Senkin voisi suorittaa R:llä, joskaan ei yhdellä komennolla. Kiinnostuneet voivat yrittää itse.

Mikäli olemme epävarmoja jakaumaoletuksista, tunnusluvun merkitsevyyden pystyy arvioimaan myös simuloidulla otannalla kontingessitaulukosta:

```
> chisq.test(ruoka, simulate.p.value=T)

        Pearson's Chi-square test with simulated p-value
        (based on 2000 replicates)

data:  ruoka
X-squared = 30.2696, df = NA, p-value = < 2.2e-16
```

Khi–neliön testiä voi käyttää myös tutkimaan, poikkeaako havaittu jakauma muusta odotetusta jakaumasta kuin tasajakaumasta. Ranta *et al.* (1989, Esimerkki 2.2) käsittelevät viirupöllöjen munamäärää "Lahden takamailla". Aineistossa on munamäärät $0 \ldots 7$ (muuttuja `munat`) ja niiden pönttöjen lukumäärä, josta kyseinen munamäärä on havaittu (muuttuja `viiru`):

```
> viiru <- c(4, 5, 7, 11, 29, 26, 7, 2)
> munat <- 0:7
```

Usein tällaiset lukumäärämuuttujat noudattavat Poisson–jakaumaa. Testatkaamme, kuinka käy tässä tapauksessa. Poisson–jakauman todennäköisyydet riippuvat vain munamäärän keskiarvosta:

```
> ka <- weighted.mean(munat,viiru)  ## Painotettu keskiarvo
> odotus <- c(dpois(0:6, ka), 1-ppois(6, ka))
```

Funktio `dpois` antaa tietyn munamäärän todennäköisyyden, ja `ppois` antaa vastaavan summatodennäköisyyden: Poisson–jakauman mukaan meillä voisi olla myös enemmän kuin 7 munaa, joten viimeinen termi on todennäköisyys saada ≥ 7 munaa. Havaittu jakauma ja Poisson–jakauma näyttävät erilaisilta:

```
> plot(munat,viiru, type="h",lwd=3)
> segments(munat+0.1,0, munat+0.1,sum(viiru)*odotus, lty=2)
```

Komento tuottaa "oikeaoppisen" impulssidiagrammin. "Vääräoppinen", mutta tavallisempi pylväsdiagrammin saadaan komennolla

```
>  barplot(rbind(viiru, sum(viiru)*odotus), beside=T)
```

Teoreettista jakaumaa vasten testataessa, `chisq.test` tarvitsee nimenomaan havaitut lukumäärät ja odotetut todennäköisyydet (summa = 1), joten testi on:

```
> chisq.test(viiru, p=odotus)

        Chi-square test for given probabilities

data:  viiru
X-squared = 33.4043, df = 7, p-value = 2.226e-05

Warning message:
Chi-square approximation may be incorrect in: chisq.test(viiru, p = odotus)
```

Osa odotusfrekvensseistä oli pieniä, joten saimme varoituksen. Ilmeisesti munamäärä ei kuitenkaan jakaudu poissonisesti.

6.2.2 Kolmogorovin–Smirnovin testi

Kolmogorovin–Smirnovin testisuure on odotetun ja havaitun kumulatiivisen frekvenssin erotus D. Kolmogorovin–Smirnovin testillä voi verrata kahden jatkuvan muuttujan jakaumaa keskenään ("Smirnovin testi") tai yhtä jatkuvaa muuttujaa teoreettiseen jakaumaan ("Kolmogorovin testi"). Molemmat testit voi suorittaa paketin `ctest` ohjelmalla `ks.test`. Kutsujen periaate on:

```
> ks.test(havaittu, pjakauma)
> ks.test(havaittu1, havaittu2)
```

R:ssä on suuri joukko jakaumafunktiota tyyppiä `pjakauma`, esim. `punif` (tasainen jakauma) ja `pnorm` (normaalijakauma), joita voi käyttää Biometrian kirjan esimerkkien ratkaisussa.

Ranta *et al.* (1989, Esimerkki 6.5) esittelevät tutkimusta, jossa selvitettiin, kuuleeko Ornitologi pajulinnun yhtä hyvin läheltä ja kaukaa. Talletamme etäisyydet, jolta pajulinnut on kuultu muuttujaan `tirppa`.

```
> tirppa <- c(2,4,4,8,9,16,18,19,24,25,26,32,36,41,54,55,56,66,72,77,90,96)
```

Ranta *et al.* (1989) käyttivät muuttujan `tirppa` jakajana arveltua kuuluvuuden maksimietäisyyttä 100 m, jolloin muuttujan maksimifrekvenssiksi tulee 0.96. Vertaamme havaittuja jakaumaa teoreettiseen jakaumaan, eli teemme yhden otoksen testin. Nollahypoteesin mukainen jakauma on tasajakauma, jonka R:ssä saamme funktiolla `punif`.

```
> ks.test(tirppa/100, punif)

        One-sample Kolmogorov-Smirnov test

data:  tirppa/100
D = 0.24, p-value = 0.1585
alternative hypothesis: two.sided
```

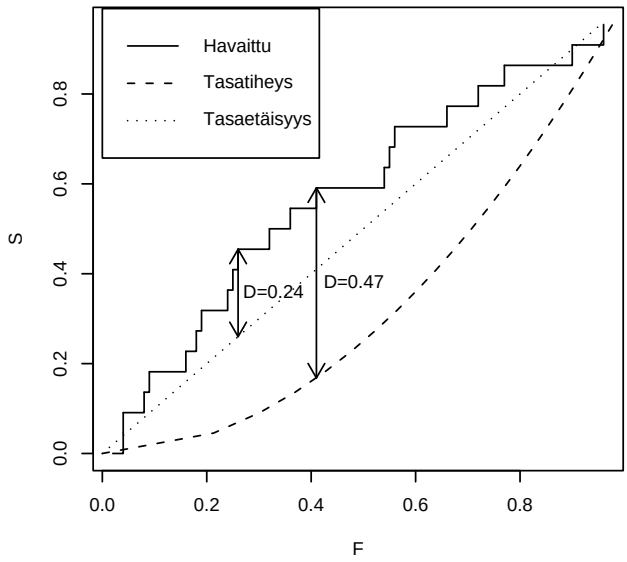
Testisuure $D = 0.24$ ei ole "tilastollisesti merkitsevä", joten aineisto ei tarjoa riittävästi näyttöä tasajakaumahypoteesia vastaan (eli tavanomaisten tilastotieteen alkeisesitysten mukaan "nollahypoteesi jää voimaan"). Testisuure poikkeaa Biometrian esimerkistä (Ranta *et al.*, 1989, Esimerkki 6.5). Tämä johtuu siitä, että Ranta *et al.* (1989) määrittelevät tasavälisen vertailumuuttujan S vaihteluväliksi $1/n \ldots 1$, kun taas R käyttää väliä $0 \ldots (1-1/n)$ — minulla ei ole aavistustakaan, kumpi on oikea tapa.

Valitettavasti testasimme virheellistä nollahypoteesia: Ranta *et al.* (1989) kirjoittavat: "Tutkija tiesi, että pajulinnun populaatiotiheys on sama tutkimusalueen eri osissa ... Nollahypoteesin mukaan kuulohavainnot jakautuvat tasaisesti." Jos tiheys on tasainen, kuulojakauman *ei* pidä olla tasainen, vaan kaukaa pitäisi kuulua useampi lintu, sillä kaukana on suurempi pinta-ala kuin lähellä. Kuulohavaintojen pitäisi siis jakautua kuten pinta-alan eli suhteessa etäisyyden neliöön. Tällöin saamme:

```
> ks.test((tirppa/100)^2, punif)

        One-sample Kolmogorov-Smirnov test

data:  (tirppa/100)^2
D = 0.4683, p-value = 0.0001291
alternative hypothesis: two.sided
```

Päätelmämme on siis erilainen: lähellä olevat linnut kuullaan todennäköisemmin kuin kaukana olevat, mikä yhtenee arkikokemukseemme äänisignaalin vaimenemisesta.

Lopuksi piirrämme kuvan 3. Kuvassa käytetään askelfunktiota havaitulle jakaumalle: arvo muuttuu askeleittain juuri havainnon kohdalla. Komentojen lyhentämiseksi tällä kertaa lasketaan myös



Kuva 3: Kolmogorin–Smirnovin testi: pajulinnun äänen havaitseminen eri etäisyyksiltä oikeaa ja väärää nollahypoteesia vasten.

apumuuttujat `F` (havaittu jakauma)[6] ja `S` (odotettu jakauma):

```
> F <- tirppa/100
> S <- 0:(n-1)/n    ## Kirja käyttää S <- 1/n:n
> plot(F,S,type="s")
> lines(S,S,lty=3); lines(sqrt(S),S,lty=2)
> legend(0,0.99,legend=c("Havaittu","Tasatiheys","Tasaetäisyys"),lty=(1:3))
> arrows(0.26,0.26, 0.26,0.4545, code=3,length=0.1)
> arrows(0.41,0.1681, 0.41,0.5909, code=3,length=0.1)
> mp <- locator()   ## Osoitetetaan hiirellä tekstin paikat
> text(mp,c("D=0.24","D=0.47"), adj=c(0,1))
```

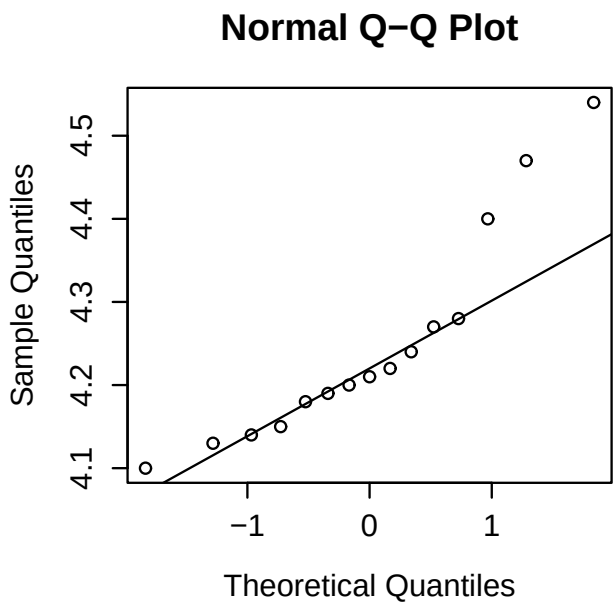
Nuolten koordinaatit jouduimme laskemaan erikseen seuraamalla kirjan esimerkkiä (Ranta *et al.*, 1989). Nuolten koordinaatit löytyivät tutkimalla tuloksia komennoista:

```
> cbind(F,S,F-S,F-S-1/length(F))
> cbind(F^2,S,F^2-S,F^2-S-1/length(F))
```

Toisena esimerkkinä Ranta *et al.* (1989, Esimerkki 6.6) käsittelevät aineistoa, jossa oli punnittu koko joukko männyn siemeniä ja haluttiin tietää "noudattivatko ne normaalijakaumaa". Kolmogorovin–Smirnovin testissä käyttämämme jakaumafunktio `pnorm` on jakautunut $\sim N(0,1)$, joten meidän on tehtävä z–muunnos, mikä tapahtuu helpoimmin funktiolla `scale` (sivu 8):

```
> siemen <- c(4.1, 4.13,4.14,4.15,4.18,4.19,4.2,4.21,4.22,4.24,
+             4.27,4.28,4.4,4.47,4.54)
> ks.test(scale(siemen), "pnorm")

        One-sample Kolmogorov-Smirnov test

data:  scale(siemen)
```

[6] Muuttuja `F` kannattaa poistaa (`rm(F)`) kun sitä ei enää tarvita: muutoin komentojen määritteet (`vipu=F`) epäonnistuvat salaperäisesti.



Kuva 4: Siemenpainojen jakauma komennolla `qqnorm(siemen); qqline(siemen)`

```
D = 0.2012, p-value = 0.5784
alternative hypothesis: two.sided
```

Saamamme testisuure $D = 0.201$ ei ole "merkitsevä" eli meillä ei tämän testin mukaan ole riittävästi näyttöä "hylätä" normaalijakaumaa. Sekä testisuure että johtopäätös poikkeavat Biometrian kirjasta, mutta tämä johtuu kirjassa olevassa laskuvirheestä.[7] Graafisesti tarkasteltuna empiirinen jakauma vaikuttaa ei-normaaliselta (kuva 4).

Ilmeisesti Kolmogorovin–Smirnovin testi ei ole kovin voimakas tässä tapauksessa. Sen sijaan normaalijakautuvuuden testaamiseen kehitetty Shapiron–Wilkin testi pitää jakaumaa melko varmasti epänormaalina:

```
> shapiro.test(siemen)

        Shapiro-Wilk normality test

data:  siemen
W = 0.8682, p-value = 0.03171
```

Shapiron–Wilkin testi onkin tehokkaampi kuin Khi–neliön tai Kolmogorovin–Smirnovin testit, jos meillä todella on tarve testata jakauman normaalisuutta.

Kaikkia jakaumatestejä käytettäessä on syytä muistaa, että ne ovat "asymptoottisesti merkitseviä" eli aineiston koon kasvaessa ne pystyvat havaitsemaan vähäisetkin poikkeamat jakaumaoletuksesta. Useimmiten meillä ei ole erityistä syytä olettaa, että jakauma olisi normaalinen, joten testaamme lähinnä otoskoon riittävyyttä havaitsemaan poikkeavuudet. Jos saamme tuloksen, että havaittu jakauma ei poikkea "merkitsevästi" normaalisesta, se ei suinkaan osoita että jakauma "noudattaa normaalijakaumaa", ainoastaan että se voisi olla normaalinen. Mitä ilmeisimmin on olemassa suuri joukko muitakin jakaumia, joista havaittu jakaumamme ei poikkea "merkitsevästi" ja riittävän suurella aineistolla se poikkeaisi niistä kaikista.

[7] $0.800 - 0.599 \neq 0.401$: Ranta *et al.* (1989, Esim. 6.6, rivi 12).

7 Lineaariset mallit

Sekä regressioanalyysi että varianssianalyysi ovat erikoistapauksia *lineaarisista malleista*, joissa selitettävän muuttujan eli vastemuuttujan (*response variable*) oletetaan olevan jatkuva ja välimatka-asteikollinen. Ne eroavat toisistaan lähinnä selittävien tekijöiden ominaisuuksien perusteella:

- Jos selittävät tekijät ovat jatkuvia, puhumme *regressioanalyysistä*.
- Jos selittävät tekijät ovat luokkamuuttujia, puhumme *varianssianalyysistä*
- Jos selittävät tekijät sisältävät kumpiakin, puhumme usein *kovarianssianalyysistä*.

Tilastotieteen oppikirjoissa regressioanalyysi ja varianssianalyysi esitellään usein erillisinä menetelminä (esim. Ranta *et al.*, 1989). Tämäkin on perusteltua, sillä etenkin manuaaliset laskutoimitukset ovat hyvin erilaisia. Lisäksi tulosten perinteiset esitystavat eroavat, sillä kumpikin menetelmä on kehittynyt pitkään erikseen. Tietokoneella laskettaessa sen sijaan kumpikin voidaan laskea hyvin samalla tavoin ja esittää yhtenäisesti. Joissakin tilasto-ohjelmissa (esim. SAS, SPSS) lineaarisia malleja nimitetään *yleisiksi lineaarisiksi malleiksi* ("*general linear models*") ja lyhennetään GLM. Koska ne ovat nykyisessä katsannossa vain erityistapauksia vielä yleisemmästä malliperheestä nimeltään *yleistetyt lineaariset mallit* ("*generalized linear models*"), R:ssä niitä nimitetään vain lineaarisiksi malleiksi, ja ne voidaan sovittaa funktiolla `lm`. Funktio (ja nimilyhenne) `glm` on varattu yleistetyille lineaarisille malleille, jonka erikoistapauksia ovat juuri mainitut jatkuvan vastemuuttujan lineaariset mallit mutta lisäksi kaksiarvoisten vasteiden ja lukumäärävasteiden analyysin tarkoitetut regressiomallit.

7.1 Mallilause: yksinkertaisin tapaus

Lineaarisen regressiomallin systemaattinen merkitään tilastokirjoissa tyyliin

$$E(y) = b_0 + b_1 x \tag{4}$$

Toisin sanoen jatkuvan *vastemuuttujan* y odotusarvo ($E(y)$) määräytyy selittävän muuttujan x (välimatka-asteikollinen tai kaksiarvoinen) lineaarisena funktiona: odotusarvo saadaan laskemalla yhteen *perustasokerroin* (*intercept*) b_0 ja muuttujan x arvo kerrottuna *kulmakertoimella* (*slope*) b_1. Mallin satunnainen osa kuvaa sitä, mikä on vastemuuttujan arvojen vaihtelu odotusarvonsa ympärillä kullakin x:n tasolla erikseen. Usein oletetaan, että tämä vaihtelu noudattaa normaalijakaumaa odotusarvolla 0 ja tuntemattomalla vakiovarianssilla σ^2.

Voimme esittää yhtälön 4 mallin odotusarvo-osan kahdella vaihtoehtoisella tavalla R:ssä:

```
lm(y ~ x + 1)
  lm(y ~ x)
```

Suluissa oleva lause voidaan lukea "`y` riippuu (`~`) `x`:stä". Sulkuja edeltävä `lm` taas kertoo että tämä riippuvuus on lineaarista eli sellaista kuin yhtälö 4 määrittää — myös muunlaisia riippuvuuksia voi määritellä, vaikka tällaisiin kehittyneempiin malleihin ei tässä oppaassa päästäkään. Ensimmäisen lauseen termi "`1`" esittää vakiokertoimen b_0 eksplisiittisesti, jälkimmäinen vain implisiittisesti eli oikealla puolella on yksinäinen `x`. Koska useimmiten ei ole mitään järkevää syytä jättää vakiotermiä b_0 pois, molemmat muotoilut määrittelevät saman mallin. Jos haluaisimme *todellakin* jättää vakiotermin pois eli pakottaa regressiosuoran origon kautta, meidän pitäisi kirjoittaa `y ~ x - 1`.

Malliyhtälöissä käytetyn "madon" eli *tilde*n (`~`) kirjoittaminen voi tuottaa vaikeuksia joillain koneilla. Se on näet usein määritelty kuuroksi merkiksi eli se ei näy välittömästi ruudulla vaan odottaa, seuraako jokin kirjain, jonka päälle se voisi asettua (kuten vironkielen "õ"). Välilyöntinäppäimen painallus riittää usein tilden tulostumiseen. Joissain tapauksissa saattaa olla välttämätöntä konfiguroida kone uudelleen, jotta "`~`" saadaan toimimaan.

Mallilause on lineaaristen mallien sydän. Sen avulla pystyy yllättävän helposti määrittelemään hankalankin tuntuisia malleja, itse asiassa paljon helpommin kuin graafisten kilkkeiden avulla.

7.2 Yksinkertainen regressio

Ranta *et al.* (1989, Esimerkki 10.1) käyttävät yksinkertaisen regressioanalyysin esimerkkinä aineistoa, jossa tutkitaan ravinnon määrän vaikutusta vesiliskon lisääntymiseen. Kohdemuuttujana on vesiliskon tuottamien munien määrä (muuttuja `munat`), selittävänä tekijänä vesiliskojen ruokintakerralla saama surviaistoukkien lukumäärä (muuttuja `ruoka`). Näin pieni aineisto on helppo tallentaa itsekin suoraan:[8]

```
> munat <- c(4, 1, 6, 2, 5, 11, 7, 11, 9, 15)
> ruoka <- seq(along=munat)
```

Sovitamme regressiomallin ja pyydämme yhteenvedon tuloksista (olen poistanut osan `summary`n riveistä kaikissa esimerkeissä):

```
> lisko.reg <- lm(munat ~ ruoka)
> summary(lisko.reg)

Call:
lm(formula = munat ~ ruoka)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.400      1.750   0.229  0.82489
ruoka          1.218      0.282   4.320  0.00254

Residual standard error: 2.561 on 8 degrees of freedom
Multiple R-Squared:   0.7,      Adjusted R-squared: 0.6625
F-statistic: 18.67 on 1 and 8 degrees of freedom,       p-value: 0.002545
```

Regressioanalyysin tulos `lisko.reg` on luokan `lm` olio. Komento `summary` tuntee luokan, ja osaa esittää tärkeimmät tiedot. Komento `print` (jota kutsuu epäsuorasti nimen `lisko.reg` kirjoittaminen) antaisi niukemmat tiedot. Komennolla `attributes(lisko.reg)` saamme luettelon olion attribuuteista. On myös joitain erityismenetelmiä eristää joitain tiettyjä luokan attribuutteja. Esim. `coef(lisko.reg)` antaa pelkät regressiokertoimet. Ainoan selittävän tekijän `ruoka` regressiokerroin on 1.22: näin paljon munamäärä kasvaa kun ruokintaa lisätään yhdellä surviaisella. R tulostaa myös kertoimien keskivirheen (`Std. Error`) ja t-arvon, joka on estimaatti jaettuna keskivirheellään. Kertoimen merkitsevyyden arviointi perustuu t-testiin (§6.1). Ruokinta lisää merkitsevästi jälkeläistuotantoa ($p = 0.0025$), mikä ei ole järisyttävä tieto. R saattaa tulostaa myös tähdet (`*`) "merkitsevien" kertoimien jälkeen, mutta olen tarkoituksella poistanut tämän toiminnon (katso `options`), koska tiukat tähtirajat ovat arveluttavia — onko $p = 0.049$ todellakin yhden tähden arvoinen, mutta $p = 0.051$ arvoton?

Vakiotermin `(Intercept)` merkitys ja merkitsevyys on usein epäselvä ja niin myös tässä tapauksessa. Vakiotermi ilmoittaa suoran arvon pisteessä $x = 0$ ja niinpä vakio b_0 on mielekäs vain jos piste $x = 0$ on jotenkin erikoinen ja mielekäs. Origon siirto muuttaa sekä vakiotermiä että sen merkitsevyyttä. Tällä kertaa vakiotermi on ekstrapoloitu sillä yksi toukka oli pienin ruokintamäärä. Vakiotermi ei ole erityisen kiinnostava, sillä se ilmoittaisi jälkeläistuoton silloin kun eläimet eivät saisi lainkaan ruokaa, mikä ei varmaankaan ole kestävä tilanne. Toisaalta on mahdollista, että alimmat ruokintamäärät eivät olleet riittäviä vaan eläimet näkivät jo nyt nälkää ja lisääntyivät vain varantojensa kustannuksella. Mikäli tietäisimme riittävän ravintomäärän kriittisen rajan, voisimme siirtää x-muuttujan nollapisteen tuohon rajaan ja kenties pakottaa vakiotermin nollaksi tuossa pisteessä, mutta meillä ei ole mitään perusteltua syytä pakottaa regressiota nykyisen origon kautta.

[8] Se on kuitenkin paketissa `rekola` aineistona `vesilisko`

Tulostuksen viimeinen rivi kertoo koko mallin merkitsevyyden F-tunnusluvulla eli varianssianalyysillä arvioituna. Koska meillä on vain yhden selittäjän regressio, koko mallin merkitsevyys F-tunnusluvun perusteella on sama kuin ainokaisen selittäjän regressiokertoimen merkitsevyys t-tunnusluvulla arvioituna.

Varianssianalyysi on vielä edessä päin, mutta tässä oppaassa oletetaan, että tilastotieteen perusteet ovat takana päin. Niinpä ei pitäisi olla uutinen, että mallin sopivuutta aineistoon kuvaa *jäännösneliösumma* ("*Residual Sum of Squares*") ja mallin *vapausasteiden* määrä on havaintojen määrä miinus estimoitujen parametrien määrä. Estimoitujen parametrien lisääntyessä vapausasteet kuluvat ja jäännösneliösummat (todennäköisesti) pienenevät. Jos testattavan ja nollamallin jäännösneliösummat ovat SS_1 ja SS_0 ja vapausasteet df_1 ja df_0, niin F-tunnusluku vapausasteilla $df_0 - df_1, df_1$ on:

$$F_{df_0-df_1,df_1} = \frac{(SS_0 - SS_1)/(df_0 - df_1)}{SS_1/df_1} \tag{5}$$

R-funktio `anova` tekee automaattisesti nämä laskut ja arvioi F-tunnusluvun merkitsevyyden kun sille annetaan kaksi luokan `lm` oliota. Sovitamme ensin mallin, jossa on vain vakiotermi ja sitten vertaamme tätä malliimme `lisko.reg`, jossa on myös kulmakerroin tekijälle `ruoka`:

```
> lisko.null <- lm(munat ~ 1)  ## Vain vakiotermi
> anova(lisko.null, lisko.reg)
Analysis of Variance Table

Model 1: munat ~ 1
Model 2: munat ~ ruoka
  Res.Df Res.Sum Sq Df  Sum Sq F value   Pr(>F)
1      9    174.900
2      8     52.473  1 122.427  18.665 0.002545
```

Mallien jäännösneliösummat olivat $SS_0 = 174.9$ ja $SS_1 = 52.5$. Sijoitettuna yhtälöön 5 saamme:

$$F_{1,8} = \frac{(174.9 - 52.473)/(9 - 8)}{52.473/8} = \frac{122.427/1}{52.473/8} = 18.67,$$

joka on sama tulos kuin suoraan regressionalyysin `summary`ssa.

Ylläolevan komennon `anova` tulostuksesta saamme myös muut `summary`n alarivien tulokset. Jäännöskeskihajonta, jota R kutsuu nimellä "`Residual standard error`", vastaa muuten tavallista keskihajontaa, mutta se on laskettu regressiosuoran eikä y-muuttujan koko aineiston keskiarvon suhteen. Jäännöskeskihajonta on oletetun mallin teoreettisen virhekeskihajonnan σ estimaattori. Se saadaan `anova`–taulukosta laskuilla

$$s_{\text{resid}} = \sqrt{\frac{SS_1}{df_1}} = \sqrt{\frac{52.473}{8}} = 2.56$$

"`Multiple R-Squared`" on suomeksi *determinaatiokerroin* eli "yhteiskorrelaatiokertoimen neliö", ja se saadaan niin ikään `anova`-taulukosta:

$$R^2 = 1 - \frac{SS_1}{SS_0} = 1 - \frac{52.437}{174.9} = 0.70$$

R^2 tunnetaan myös nimellä "selitysaste": se kuvaa kuinka suuren osan vastemuuttujan arvojen varianssista kokonaisneliösummasta (SS_0) regressiosuora kykenee ao. aineistossa "selittämään". "`Adjusted R-squared`" on samantapainen, mutta sitä on rangaistu ("penalisoitu") estimoitujen parametrien määrällä.

Näistä alarivien tuloksista suosituin ja yliarvostetuin on determinaatiokerroin eli R^2. "Selitysaste" ei esimerkissämme ole erityisen mielekäs, sillä jälkeläistuotannon varianssi on syntynyt keinotekoisessa kokeessa jossa on systemaattisesti varioitu ruoan määrää. Mikäli ruokintamäärän varianssi olisi ollut vähäisempi (vähemmän ruokintatasoja tai enemmän toistoja keskimmäisillä tasoilla), myös munatuotannon varianssi olisi alentunut ja yhtä hyvällä regressiolla "selitysaste" olisi alentunut. Myös havaintoaineistoissa tutkittavan muuttujan vaihtelun kasvattaminen on usein helpoin tapa saada hyviä "selitysasteita". Regressiomallin antaman ennusteen hyvyyttä kuvaa paremmin absoluuttinen vaihtelu regressiosuoran ympärillä eli residuaalivaihtelu s_{resid}: jos vaihtelu on vähäistä, regressio on ennustusvoimainen riippumatta "selitysasteesta". Esimerkissämme $R^2 = 0.7$, mikä vaikuttaa korkealta, mutta $s_{\mathrm{resid}} = 2.56$, mikä ei minusta vaikuta kovinkaan tyydyttävältä.

Kenties vielä paremman kuvan mallimme toimivuudesta saamme tarkastelemalla regressiosuoran ja ennusteiden luottamusrajoja: rajoja, joiden sisällä oikea regressiosuora tai ennustettu yksittäinen arvo on esim. 95 % todennäköisyydellä. Ne voidaan laskea komennolla `predict.lm`, kuten alla tehdään kuva piirrettäessä.

Luokan `lm` olioille on oma `plot`-komento, joka tuottaa lähinnä diagnostiikkaan tarkoitettuja kuvia. Koska meillä on vain yksi selittävä tekijä, tarkastelemme ennemmin perinteisiä regressiokuvia; monimutkaisemmissa malleissa tutkimme oletuskuvia. Peruskuva syntyy komennoilla:

```
>  plot(ruoka,munat,xlab="Ravinto (toukkaa/kerta)",ylab="Munien tuotto")
>  abline(lisko.reg)          # Regressiosuora
>  abline(h=mean(munat),lty=2)
>  abline(v=mean(ruoka),lty=2)
```

Viivoja kuvaan lisäävä `abline` osaa eristää tuloksen regressiokertoimet ja piirtää regressiosuoran. Kaksi viimeistä `abline`-komentoa lisää viivat muuttujien keskiarvon kohdalle muistuttamaan, että regressiosuora kulkee kummankin muuttujan keskiarvon kautta.

Tarvitsemme vielä luottamusrajat. Ne on laskettava erikseen komennolla `predict.lm`:

```
> lisko.ci <- predict(lisko.reg, interval="confidence")
```

Tuloksena on matriisi, jossa on kolme saraketta: ennustettu arvo (`fit`), alempi luottamusraja (`lwr`) ja ylempi raja (`upr`). Oletuksena on käyttää 95 % rajoja. Selittävät muuttujat ovat valmiiksi oikeassa järjestyksessä, joten voimme lisätä kuvaan rajat komennoilla:

```
>  lines(ruoka,lisko.ci[,"lwr"], lty=2)
>  lines(ruoka,lisko.ci[,"upr"], lty=2)
```

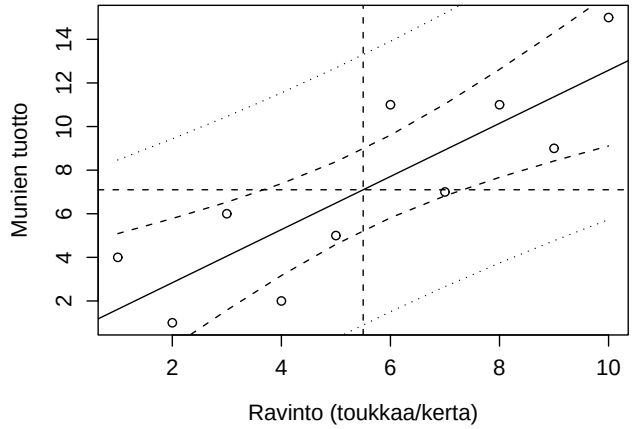
Mikäli "`[,"lwr"]`" näyttää merkilliseltä, parasta kerrata matriisit (§3.6).

Nämä ovat *regressiosuoran* luottamusrajat: 95 % todennäköisyydellä regressiosuora on rajojen sisällä. Kuten näkyy, luottamusrajat ovat kapeimmillaan selittävän tekijän keskiarvon kohdalla (kuva 5). Mikäli haluamme käyttää regressiosuoraa ennustamaan yksittäisen vesiliskon munatuottoa, on käytettävä vielä laajempia rajoja, eli *ennusteen* luottamusväliä. Tähän saakka käyttämämme rajat esittävät vain suoran epävarmuutta, mutteivät yksittäisten arvojen poikkeamista regressiosuorasta. Laskemme vielä ennusteen luottamusrajat ja lisäämme ne kuvaan, jolloin saamme lopulliseksi tulokseksi kuvan 5. Paitsi komentoa `lines`, voimme käyttää myös komentoa `matlines`, joka lisää molemmat rajat yhtaikaa.

```
>  lisko.pi <- predict(lisko.reg, interval="prediction")
>  matlines(ruoka, lisko.pi[,c("lwr","upr")], lty=3, col="black")
```

Komento `matlines` vaihtaa väriä ja viivatyyppiä matriisin jokaiselle sarakkeelle, joten jouduimme estämään tämän määritteillä `lty` ja `col`. Ennusteen luottamusrajat ovat todella väljät eikä mallimme näytä olevan kovinkaan käyttökelpoinen mikäli tarkoitus on ennustaa yksittäisen vesiliskon tuottama munamäärä. Esim. viidellä toukalla per ruokintakerta, 95 % vesiliskoista tuottaa 0.3 . . . 12.7 munaa:

```
> lisko.pi[5,]
       fit        lwr        upr
```



Kuva 5: Yksinkertainen regressioanalyysi sekä regression ja ennusteen 95 % luottamusrajat. Pysty- ja vaakasuorat katkoviivat osoittavat kummankin muuttujan keskiarvoja.

```
6.4909091  0.2882827 12.6935355
```

Ranta *et al.* (1989) esittelevät vain regressiosuoran luottamusrajat, joita he harhaanjohtavasti nimittävät "ennusteen luottamusrajoiksi".

7.3 Monen selittävän tekijän regressio

Yhden selittävän tekijän regressioyhtälö (4) on helposti laajennattavissa monen selittävän tekijän malliksi:

$$E(y) = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_p x_p \tag{6}$$

Tämäkin on lineaarinen malli, vaikka tulokset saattavat joskus olla käyriä, kuten

$$E(y) = b_0 + b_1 x + b_2 x^2$$

joka on paraabeli. Lineaarisuus regressiomallissa tarkoittaa, että malliyhtälössä on vakiotermin lisäksi vain kerroin × tekijä -tyyppisiä termejä. Hienommin sanottuna, malli on lineaarinen parametrien suhteen (*linear in parameters*).

Yhtälöä 6 vastaa R:ssä mallilauseke (vakiotermin "`1`" voi tietysti jättää kirjoittamatta näkyviin):

```
lm(y ~ 1 + x1 + x2 + ... + xp)
```

Monen selittävän muuttujan regression sovittaminen R:ssä on aivan yhtä helppoa kuin mallilausekkeen kirjoittaminen. Tässä suhteessa ei paljonkaan uutta. Monen selittäjän malleihin liitty kuitenkin ongelmia tulkinnassa, esitystekniikassa ja mallin rakentamisessa, joten niitä on tarkasteltava erikseen.

Biometrian kirjassa käsitelty esimerkki (Esimerkki 10.8 Ranta *et al.*, 1989, aineisto `plutakko` paketissa `rekola`) tutkii kalliolammikoiden perustuotantoa, oletettavasti Tvärminnessä. Vastemuuttujana on hiilen yhteytys μg dm^{-3} h^{-1} (muuttuja `Fotosynt`), mahdollisina selittäjinä pikoplanktonin (muuttuja `Pikopl`) ja suuremman planktonin (muuttuja `Plankt`) klorofylli-*a* sekä levien käyttämättä jättämä NH_4 (muuttuja `NH4`) ja PO_4 (muuttuja `PO4`), kaikki selittäjät μg dm^{-3}.

Biometrian kirjassa tarkastellaan kahta mallia (Ranta *et al.*, 1989). Ensimmäisessä selittäjinä ovat planktonfraktioiden klorofyllitiheydet:

```
> data(plutakko)
> plut.lm <- lm(Fotosynt ~ Pikopl + Plankt, data=plutakko)
> summary(plut.lm)

Call:
lm(formula = Fotosynt ~ Pikopl + Plankt, data = plutakko)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.18388    0.26309   0.699   0.4941
Pikopl       0.50003    0.17562   2.847   0.0111
Plankt       0.45827    0.08723   5.254 6.47e-05

Residual standard error: 0.7041 on 17 degrees of freedom
Multiple R-Squared: 0.688,      Adjusted R-squared: 0.6513
F-statistic: 18.75 on 2 and 17 degrees of freedom,      p-value: 5.013e-05
```

Molemmat tekijät ovat hyvin merkitseviä ja niiden kertoimet keskenään melkein yhtäsuuria.

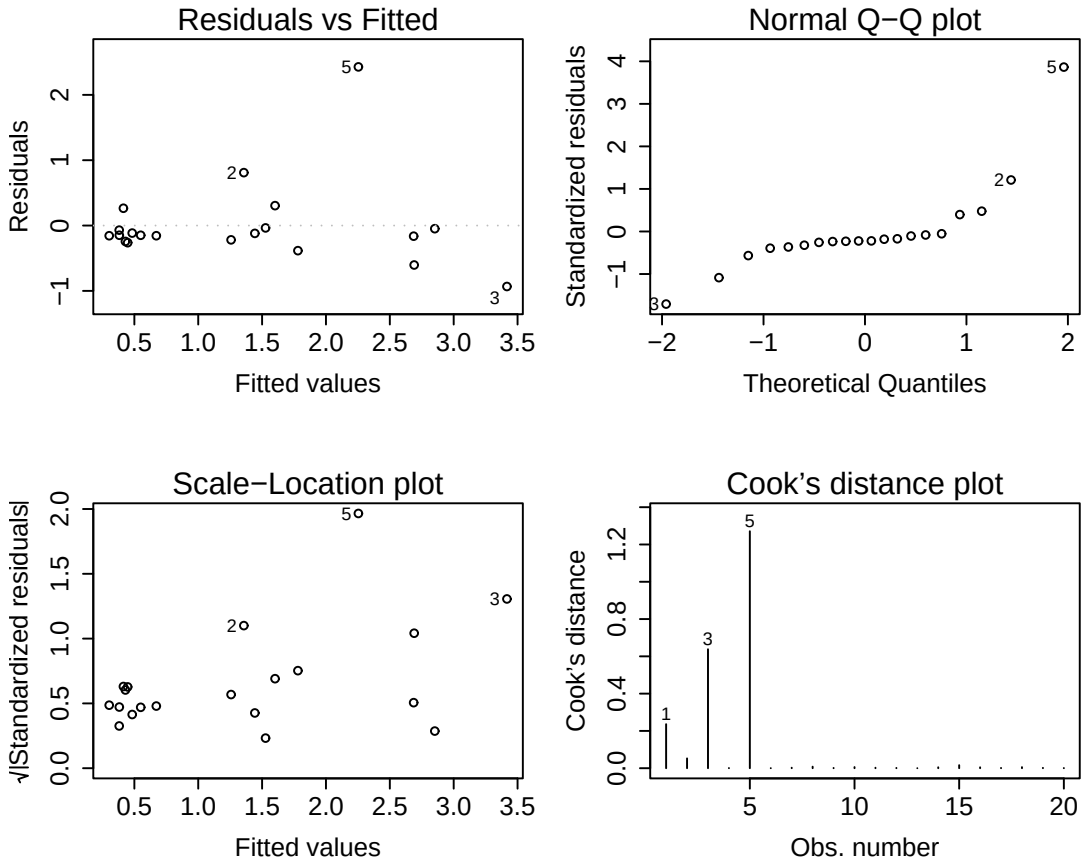
Toisessa mallissa käytettiin kaikkia selittäviä tekijöitä, joten R:n mallilausekkeen voi mukavasti lyhentää:

```
> plut.full <- lm(Fotosynt ~ . , data=plutakko)
> summary(plut.full)

Call:
lm(formula = Fotosynt ~ ., data = plutakko)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.15390    0.65204   0.236  0.81660
Pikopl       0.52813    0.20895   2.528  0.02321
Plankt       0.49325    0.10516   4.690  0.00029
NH4          0.03771    0.11255   0.335  0.74226
PO4         -0.15870    0.21136  -0.751  0.46436

Residual standard error: 0.7347 on 15 degrees of freedom
Multiple R-Squared: 0.7002,     Adjusted R-squared: 0.6203
F-statistic:  8.76 on 4 and 15 degrees of freedom,      p-value: 0.0007443
```

Ravinteet eivät ole merkitseviä, mutta planktonfraktiot ovat. Planktonfraktioiden kertoimet muuttuivat ja kertoimien p–arvot jopa huononivat. Myös `anova` on sitä mieltä, että ravinnetermien lisääminen ei merkitsevästi parantanut mallia:

```
> anova(plut.lm,plut.full)
Analysis of Variance Table

Model 1: Fotosynt ~ Pikopl + Plankt
Model 2: Fotosynt ~ Pikopl + Plankt + NH4 + PO4
  Res.Df Res.Sum Sq Df Sum Sq F value Pr(>F)
1     17     8.4275
2     15     8.0973  2 0.3302  0.3058  0.741
```

`anova` testaa nyt kumpaakin ravinnetekijää yhtaikaa, joten vapaustasteiden muutos on 2. Tilastolliselta kannalta meillä ei ole mitään syytä pitää mallissa mukana ravinnetekijöitä. Meillä ei ole siihen mitään syytä myöskään ekologiselta kannalta: ravinnepitoisuudet on mitattu *vedestä* eli ne



Kuva 6: Lineaarisen mallin tuloksen diagnostinen oletuskuva (aineisto `plutakko`)

ovat kasvien *käyttämättä* jättämiä ravinteita, joiden ei voi odottaakaan selittävän fotosynteesiä.

Tarkastelemme perusteellisemmin ensimmäistä, vain planktonfraktiot sisältävää mallia. Koska meillä on kaksi selittävää tekijää, emme voi esittää tuloksia yksinkertaisena regressiokuvaajana, jossa x-akselilla on selittävä tekijä. Sen sijaan voimme tutkailla luokan `lm` olion oletuskuvaa (kuva 6):

```
> par(mfrow=c(2,2))       ## 4 osakuvaa
> plot(plut.lm)
```

Komento `plot.lm` tuottaa neljä diagnostista kuvaa, joita voi käyttää mallin arviointiin:

1. *Residuals vs Fitted*: Regression residuaalit vastaan ennustetut arvot. Mikäli malli sopii, residuaalit muodostavat tasaisen vyön suoran *Residuals* = 0 ympärille. Näin ei käy vaan residuaalit tuntuvat taipuvan alaspäin korkeammissa sovitetuissa arvoissa. Tämä viittaa siihen, että yksinkertainen lineaarinen mallimme ei sovi, vaan meidän pitäisi tutkia muita muotoja. Lisäksi hajonta saattaa hieman kasvaa kohti korkeampia ennustettuja arvoja. Meillä näkyy myös olevan muutama hyvin poikkeava arvo; havainnot 2, 3 ja 5 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 2, 3 ja 5 käyttäytyvät huonosti.

3. *Scale–Location plot*: Kuten osakuva 1, mutta nyt näytetään vain poikkeaman suuruus,

ei suuntaa. Tämä kuva osoittaa selkeämmin kuin osakuva 1 varianssin kasvun suurissa ennustetuissa arvoissa. Etenkin havainnot 2, 3 ja 5 käyttäytyvät huonosti.

4. *Cook's distance plot*: Suuri Cookin etäisyys tarkoittaa, että yksittäisellä pisteellä on suuri vaikutus regressiosuoraan. Etenkin pisteet 1, 3 ja 5 ovat huolestuttavia.

Näissä kuvissa voisi raakaresiduaalien sijaan käyttää *ulkoisesti studentoituja* (*externally studentized*) residuaaleja (katso `help(rstudent)`), jotka ovat mm. herkempiä paljastamaan oudokkeja (*outlier*).

Kaikki diagnostiset kuvat osoittivat mallimme jossain määrin huonosti sopivaksi. Havainnot 3 ja 5 olivat poikkeavia kaikissa osakuvissa ja 2 useimmissa. Sangen tavallinen ratkaisu on poistaa poikkeavat havainnot aineistosta ja koettaa uudelleen. Meillä saattoi kuitenkin olla muitakin huolen aiheita kuin kolme outoa havaintoa (mikä on 12 % aineistostamme), joten tutkimme mallia hieman paremmin.

Planktonfraktioiden kertoimien erot olivat hyvin pienet verrattuna keskivirheisiinsä, joten voisimme kokeilla näiden kertoimien pakottamista identtiseksi. Käytännössä tämä tapahtuu laskemalla yhteen muuttujat `Pikopl` ja `Plankt`. Tämä on itse asiassa myös biologisesti mielekästä: klorofyllihän yhteyttää, eikä ratkaisevaa välttämättä ole minkäkokoisen levän sisällä yhteyttävä rakenne on (itse asiassa planktonin jako kokofraktioihin on tyypillistä patriarkaatin ekologiaa, missä luonto pakotetaan tutkimuslaitteen asettamiin rajoihin). Ensimmäinen osakuva (kuva 6) viittasi myös käyräviivaiseen suhteeseen. Samaan viittaa muuttujien sirontakuvioiden tarkastelu:

```
> pairs(plutakko, panel=panel.smooth)
```

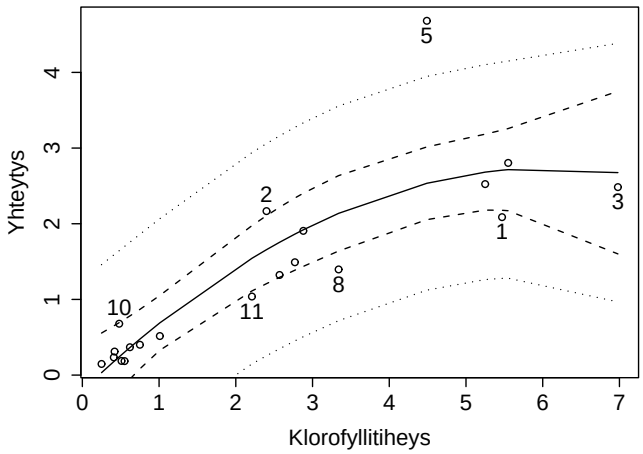
Tämän komennon kuvat (joita en näytä) viittaavat siihen, että voimme mahdollisesti aproksimoida fotosynteesin ja klorofyllitiheyden suhdetta toisen asteen yhtälöllä. On aivan varmaa, että paraabeli ei ole oikea suhde, mutta se on sopiva alustavaksi ratkaisuksi.

R:ssä on funktio `poly`, jonka avulla saamme nopeasti toisen asteen polynomin. Sen sijaan yhteenlaskun sijoittaminen mallilausekkeeseen vaatii pienen tempun, sillä mallilauseessa "`+`" tarkoittaa yleensä kummankin tekijän sisällyttämistä erikseen malliin. R:ssä on funktio `I`, joka *eristää* (*"isolates"*) operaattorit, joita käytetään tavanomaisessa matemaattisessa merkityksessä eikä mallitermeinä. Lopullinen mallimme on:

```
> plut.new <- lm(Fotosynt ~ poly(I(Pikopl+Plankt),2), data=plutakko)
> summary(plut.new)

Call:
lm(formula = Fotosynt ~ poly(I(Pikopl + Plankt), 2), data = plutakko)

Coefficients:
                            Estimate Std. Error t value Pr(>|t|)
(Intercept)                   1.3465     0.1407   9.567 2.95e-08
poly(I(Pikopl + Plankt), 2)1  4.3086     0.6295   6.845 2.85e-06
poly(I(Pikopl + Plankt), 2)2 -1.3090     0.6295  -2.080    0.053

Residual standard error: 0.6295 on 17 degrees of freedom
Multiple R-Squared: 0.7507,     Adjusted R-squared: 0.7213
F-statistic: 25.59 on 2 and 17 degrees of freedom,      p-value: 7.461e-06
```

Myös diagnostinen kuva (`plot(plut.new)`) on *paljon* parempi – sitä ei kuitenkaan näytetä, vaan se on tuotettava itse R-istunnossa. Aineistossa on yhä vielä yksi häirikkö (havainto 5), mutta muuten kaikki diagnostiset kuvaajat ovat paljon parempia. Toisen asteen termi ei itse asiassa ole merkitsevä perinteisten rajojen mukaan, mutta sen mukana olo tuntuu tekevän regressiostamme realistisemman mallisen.

Nyt meillä onkin itse asiassa vain yksi selittävä muuttuja, joten voimme piirtää perinteisen regressiokuvan (kuva 7):



Kuva 7: Lopullinen malli: regressiosuora, sen luottamusrajat sekä ennusteen luottamusrajat (aineisto `plutakko`)

```
> plut.ci <- predict(plut.new,interval="confidence") #  1
> plut.pi <- predict(plut.new,interval="prediction") #  2
> i <- sort.list(Pikopl+Plankt)                      #  3
> par(mfrow=c(1,1))                                  #  4
> plot(Pikopl+Plankt, Fotosynt,xlab="Klorofyllitiheys",ylab="Yhteytys")
> matlines((Pikopl+Plankt)[i], plut.ci[i,], lty=c(1,2,2), col="black")
> matlines((Pikopl+Plankt)[i], plut.pi[i,], lty=c(1,3,3), col="black")
> identify(Pikopl+Plankt, Fotosynt)                  # 8
[1]  1  2  3  5  8 10 11
```

Komentojono voi näyttää kauhistuttavalta, muttei kuitenkaan paljon poikkea aikaisemmista — nyt vain on hieman enemmän piirtämistä. Lisäksi kuva syntyi "iteratiivisesti", vaikka välivaiheet onkin yltä poistettu. Meitä kiinnostaa nyt sekä regressiokäyrän (rivi 1) että ennusteiden luottamusvälit (rivi 2). Käytämme ennustamiseen vanhaa aineistoa.[9] Sen takia meidän on järjestettävä kaikki ennusteet x-muuttujan mukaan ja rivillä 3 laskemme tähän tarvitsemamme indeksin `i`, jota käytämme myöhemmin riveillä 6 ja 7. Piirrämme vain yhden kuvan, joten poistamme edellisen osakuvamäärittelyn (rivi 4) ja piirrämme peruskuvan akseleineen ja datapisteineen (rivi 5). Sen jälkeen lisäämme sovitetut arvot sekä luottamusrajat (rivit 6 ja 7): tällä kertaa `matlines` helpottaa huomattavasti viivojen piirtoa. Komento `matlines` vaihtaa viivatyyppiä ja viivan väriä jokaiselle sarakkeelle, joten määritämme värin ja viivatyypin kullekin sarakkeelle komennon määreissä. Lopulta identifioimme joitain pisteitä: rivi 8 käynnistää interaktiivisen komennon `identify`, joka lisää havaintonumerot niihin pisteisiin, joita näpäytämme hiiren vasemmalla näppäimellä.

Vaikka teimme periaatteessa lineaarisen regression, saimme tulokseksi käyrän (kuva 7). Mallimme on silti lineaarinen, sillä se on lineaarisen regressioyhtälön erikoistapaus (yhtälö 6). Polynomimalli ei varmastikaan ole tässä tapauksessa oikea muoto: ekstrapoloitu käyrä painuisi alas pian tutkitun alueen ulkopuolella. Ilmeisestikin oikea muoto olisi jonkinlainen saturaatiokäyrä, missä fotosynteesi nousee aluksi melkein lineaarisesti, mutta korkeammassa klorofyllitiheydessä tasoittuu melko vakaaksi. Tällaiselle käyrälle olisi helppo keksiä ekologisesti mielekkäitä selityksiä, mutta pidättäydyn tästä, sillä meillä ei ole mitään perusteita tietää, onko selitys oikea. Vaikka regression luottamusvälit ovat melko kapeat, yksittäisen kalliolammikon yhteytyksen ennustaminen näyttää hyvin epävarmalta (kuva 7).

[9] R:ssä voi funktiolle `predict.lm` antaa myös uuden aineiston (`data.frame`), jossa on samannimiset selittävät muuttujat kuin `lm`–oliossa, mutta funktion `poly` kanssa voi tulla ongelmia mikäli tehdään uusi aineisto, sillä `poly` ei muodosta raakapolynomeja vaan ortogonaalisia polynomeja (Venables & Ripley, 1999).

7.4 Varianssianalyysi: perusmalli

Varianssianalyysi (ANOVA, *"Analysis of Variance"*) on lineaarinen malli, jossa vastemuuttuja on edelleen jatkuva mutta selittävät tekijät ovat luokkamuuttujia. Varianssianalyysin voi tehdä funktiolla `lm`. Perinteisesti varianssianalyysin tulokset esitetään kuitenkin toisin kuin regressioanalyysissä, jonka tuloksina ilmoitettiin regressiokertoimet, niiden keskivirheet ja t-arvot. Moniluokkaisen selittäjän jokainen luokka saa oman kertoimen ja muut tunnusluvut käytettäessä funktiota `lm`. ANOVAssa ei useinkaan haluta muuttujan jokaiselle luokalle tunnuslukuja vaan koko moniluokkaiselle muuttujalle yksi luku, joka kuvaa sen tilastollista merkitsevyyttä. Tämän vuoksi myös R:ssä on vaihtoehtoinen ANOVA-funktio nimeltään `aov`. Vastaavat tulokset saa myös sovittamalla hierarkisen sarjan `lm`-malleja ja vertaamalla niitä `anova`-funktiolla, kuten regressioanalyysin yhteydessä teimme. Funktio `aov` on oikotie tähän tarkoitukseen.

Tarkastelemme yksinkertaista varianssianalyysiä Biometrian oppikirjan esimerkillä, missä ollaan tutkivinaan istutustavan vaikutusta männyn taimien kasvuun (Esimerkit 8.1, 8.10 Ranta *et al.*, 1989, aineisto `taimet` paketissa `rekola`). Esimerkki on fiktiivinen, joten kovin pitkälle meneviä metsänhoidollisia päätelmiä ei pidä tehdä.

Aineisto on saatavissa suoraan komennolla `data(taimet)`, mutta opetuksellisista syistä katsomme kuinka aineisto (`data.frame`) on alkujaan muodostettu. Syötämme aineiston Esimerkin 8.10 (Ranta *et al.*, 1989) mukaisessa muodossa 3×5 taulukkona, jonka rivit ovat istustapoja ja kukin sarake edustaa yhtä emopuuta:

```
> pituus <- c(46,49,47,50,53,
+             52,54,58,61,55,
+             49,50,51,51,54)
> istutus <- c(rep("kenno",5), rep("ruukku",5), rep("rulla",5))
> emo <- rep(seq(1,5),3)
> istutus <- factor(istutus, levels=c("kenno","ruukku","rulla"))
> emo <- factor(emo)
> taimet <- data.frame(pituus=pituus, istutus=istutus, emo=emo)
> str(taimet)
'data.frame':   15 obs. of  3 variables:
 $ pituus : num  46 49 47 50 53 52 54 58 61 55 ...
 $ istutus: Factor w/ 3 levels "kenno","ruukku",..: 1 1 1 1 1 2 2 2 2 2 ...
 $ emo    : Factor w/ 5 levels "1","2","3","4",..: 1 2 3 4 5 1 2 3 4 5 ...
```

Faktorien `istutus` ja `emo` luomiseksi käytettiin sopivasti yhdisteltynä sarjoja (`seq`) ja toistoja (`rep`) (§3.4, sivu 8), joiden avulla pystyy näppärästi määrittelemään monimutkaisiakin koeasetelmia. Komento `factor` muuttaa alkuperäisen muuttujamme faktoreiksi. Faktorien tasot järjestetään nimen mukaan ja ilman parametria `levels` faktorin `istutus` tasot olisivat olleet aakkosjärjestyksessä; parametri varmistaa, että ne ovat samassa järjestyksessä kuin taulukossa alkujaan. Komennolla `data.frame` luomme aineistoin (§3.7, sivu 11). Aineisto on luettelon (§3.8, sivu 12) erikoistapaus, ja luettelo määritellään tyyliin `komponentti = muuttuja`. Alkuperäiset muuttujamme ja aineistomme komponentit ovat samannimisiä, minkä takia määrittely on hieman tyhmän näköinen. Lopulta komento `str` näyttää, että kaikki onnistui.

Biometrian kirjassa (Ranta *et al.*, 1989) aineisto analysoidaan erikseen yksisuuntaisena (Esimerkki 8.1) ja kaksisuuntaisena ANOVAna (Esimerkki 8.10). Käsin laskeminen onkin näille melko lailla erilaista, mutta R:ssä ainoa ero on mallilauseke. Tarkastelemme vain kaksisuuntaista analyysiä:

```
> taimet.lm <- lm(pituus ~ istutus + emo, data=taimet)
> summary(taimet.lm)

Call:
lm(formula = pituus ~ istutus + emo, data = taimet)

Coefficients:
```

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    46.000      1.528  30.114 1.60e-09
istutusruukku   7.000      1.414   4.950  0.00112
istutusrulla    2.000      1.414   1.414  0.19502
emo2            2.000      1.826   1.095  0.30520
emo3            3.000      1.826   1.643  0.13897
emo4            5.000      1.826   2.739  0.02550
emo5            5.000      1.826   2.739  0.02550

Residual standard error: 2.236 on 8 degrees of freedom
Multiple R-Squared: 0.8214,     Adjusted R-squared: 0.6875
F-statistic: 6.133 on 6 and 8 degrees of freedom,       p-value: 0.01122
```

R muodostaa moniluokkaisen tekijän nimen yhdistämällä muuttujan ja sen tason nimet. Kertoimet (`Estimate`) ovat tällä kertaa käsittelyn eroja verrattuna "vakioon" `(Intercept)` eli tässä tapauksessa `istutus`tapaan `kenno` ja `emo`puu 1. Estimaattien esitystapaa voi olla toinenkin, mutta siitä myöhemmin (§7.5). Tuloksia luetaan siten, että `emo`puun 1 `kenno`taimet olivat keskimäärin 46 cm pitkiä, `ruukku`taimet 7 cm ja rullataimet 2 cm `kenno`taimia *pitempiä*. Vastaavasti `emo`puun 2 jälkeläiset olivat 2 cm pitempiä kuin `emo`puun 1 jne. Aivan samoin kuin estimaatit, myös niiden keskivirheet, t-arvot ja näiden merkitsevyydet koskevat vertailua vakiotermiin. Tämä voi olla hyödyllistä, mutta tätä emme ehkä halunneet.

R-funktio `aov` tekee periaatteessa samat analyysit kuin `lm`, mutta antaa meille perinteisen ANOVA-taulukon:

```
> taimet.aov <- aov(pituus ~ istutus + emo, data=taimet)
> summary(taimet.aov)
            Df Sum Sq Mean Sq F value   Pr(>F)
istutus      2  130.0    65.0    13.0 0.003065
emo          4   54.0    13.5     2.7 0.108134
Residuals    8   40.0     5.0
```

Vaikutusten numeeriset arvot saa selville komennolla `model.tables` joka näyttää erot yleiskeskiarvoon nähden (tässä tapauksessa 52 cm):

```
> model.tables(taimet.aov)
Tables of effects

 istutus
 kenno ruukku  rulla
    -3      4     -1

 emo
 1  2  3  4  5
-3 -1  0  2  2
```

Funktio `aov` on vain "kääre" funktiolle `lm`. Pääasiallinen ero on, että `print` ja `summary` esittävät tulokset perinteisen varianssianalyysin tapaan eivätkä perinteisen lineaarisen mallin tyyliin.

Katsokaamme kuitenkin, miten `aov` itse asiassa toimii:

```
> taimet.ist <- update(taimet.lm, . ~ . -emo) # Poistetaan emo
> taimet.emo <- update(taimet.lm, . ~ . -istutus) # ja istutus
> anova(taimet.emo, taimet.lm)      ## istutuksen vaikutus
Analysis of Variance Table

Model 1: pituus ~ emo
Model 2: pituus ~ istutus + emo
```

```
  Res.Df Res.Sum Sq Df Sum Sq F value   Pr(>F)
1     10        170
2      8         40  2    130      13 0.003065
> anova(taimet.ist, taimet.lm)       ## emon vaikutus
Analysis of Variance Table

Model 1: pituus ~ istutus
Model 2: pituus ~ istutus + emo
  Res.Df Res.Sum Sq Df Sum Sq F value Pr(>F)
1     12         94
2      8         40  4     54     2.7 0.1081
```

Käytimme yllä funktiota `update` päivittämään perusmalliamma `taimet.lm`. Mallilausekkeen pisteet ". ~ ." ovat lyhenteitä päivitettävälle malliyhtälölle ja oikealle puolelle lisätty "-" tarkoittaa että poistetaan kyseinen tekijä. ANOVAssa tekijän "merkitsevyys" määräytyy sen mukaan, kuinka paljon tekijän *lisääminen* malliin pienentään neliösummaa (ja kuinka paljon se vastaavasti kuluttaa vapausasteita). Käsittelyn `istutus` vaikutus arvioidaan vertaamalla mallia, jossa on mukana *vain* `emo` (`taimet.emo`) malliin jossa on mukana *lisäksi* `istutus` (`taimet.lm`). Funktion `aov` tulostama perinteinen ANOVA-taulukko on vakiintunut tapa esittää monimutkaisten mallien vertailu yhdessä taulukossa ilman suoraa vertailtavien mallien rakentamista.

Funktio `aov` palauttaa olion, joka on perinyt luokan `lm` ominaisuudet, mutta lisäksi sillä on luokan `aov` ominaisuudet:

```
> class(taimet.lm)
[1] "lm"
> class(taimet.aov)
[1] "aov" "lm"
```

Esimerkiksi `plot.lm`-komentoa käytetään myös `aov`:n tulosten esittämiseen (ainakin vielä tätä kirjoitettaessa), joten `plot(taimet.lm)` ja `plot(taimet.aov)` tuottavat samat diagnostiset kuvat. Voimme vaatia myös käyttöön luokan `lm` komennot `aov`-olioille:

```
>  summary(taimet.aov)      # Anova-taulukko
>  summary.lm(taimet.aov)   # Sama kuin summary(taimet.lm)
```

7.5 Kontrastit ja estimaatit

Käytettäessä menetelmää `lm`, komennon `summary` antamat kertoimet tekijöille olivat ehkä hieman hämmentäviä (sivu 44): Itse asiassa R pystyy esittämään aivan *samat* tulokset hyvinkin eri tavoin sen mukaan kuinka olemme määritelleet *kontrastit*.

Tarkastelemme tällä kertaa vain yksinkertaista mallia

```
> lm(pituus ~ istutus, data=taimet)
```

eli käytämme vain selittäjää `istutus`. Tekijän `istutus` vaikutus arvioidaan vertaamalla tätä mallia nolla-malliin

```
> lm(pituus ~ 1, data=taimet)
```

Nollamallin pitäisi tuottaa estimaatiksi yleiskeskiarvo (52 cm), ja vaihtoehtoisen mallin estimaattien pitäisi jotenkin kuvata `istutus`tyyppien luokkakeskiarvoja:

```
> mean(pituus)
[1] 52
> tapply(pituus, istutus, mean)
 kenno ruukku  rulla
    49     56     51
```

Jos sovitamme mallin ilman vakiotermiä, saamme todellakin luokkakeskiarvot (kaikkia `summary`-listauksia on reippaasti lyhennetty):

```
Call:
lm(formula = pituus ~ istutus - 1, data = taimet)

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
istutuskenno    49.000      1.252   39.15 4.97e-14
istutusruukku   56.000      1.252   44.74 1.02e-14
istutusrulla    51.000      1.252   40.75 3.09e-14
```

Verratkaamme tätä vakiotermin kanssa sovitettuun malliin:

```
Call:
lm(formula = pituus ~ istutus, data = taimet)

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)     49.000      1.252  39.148 4.97e-14
istutusruukku    7.000      1.770   3.955  0.00191
istutusrulla     2.000      1.770   1.130  0.28061
```

Voimme helposti arvioida luokkakeskiarvot myös tästä tuloksesta (`kenno` $= 49$, `ruukku` $= 49 + 7$, `rulla` $= 49{+}2$ cm).[10] Jälkimmäisestä tuloksesta näemme kuitenkin myös käsittelyjen erot verrattuna ensimmäiseen käsittelyyn ja saamme niille myös t-tunnusluvut. Mikäli ensimmäinen käsittelymme on jollain tapaa mielekäs "kontrolli", voimme suoraan arvioida "käsittelyjen" vaikutusta. Esimerkkiaineistomme peitetarinan mukaan näin todellakin on eli `istutus kenno` on perusmenetelmä, jota vastaan muita menetelmiä verrataan. Merkitsevyystestien mukaan, `istutus ruukku` todellakin tuottaisi pitempiä taimia, mutta `rulla` ei poikkea merkitsevästi "kontrollista".

R:ssä voi valita kontrastien tyypin `options` komennolla. Yllä oleva R:n oletuskontrasti on:

```
> options("contrasts")
$contrasts
        unordered           ordered
"contr.treatment"      "contr.poly"
```

Kontrastit määritellään erikseen järjestämättömille (*unordered*) ja järjestetyille faktoreille (*ordered*). Järjestetyistä faktoreista puhumme myöhemmin (§7.9), joten nyt vain järjestämättömistä faktoreista. Oletusasetus on `contr.treatment`. Muita R:n tuntemia vaihtoehtoja ovat `contr.sum` ja `contr.helmert`:

```
> options(contrasts=c("contr.sum","contr.poly"))
> summary(lm(pituus ~ istutus, data=taimet))

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  52.0000     0.7226  71.957  < 2e-16
istutus1     -3.0000     1.0220  -2.935  0.01248
istutus2      4.0000     1.0220   3.914  0.00206
```

`contr.sum` käyttää vakiona `(Intercept)` yleiskeskiarvoa 52 cm ja kertoo kahden ensimmäisen luokan (`kenno`, `ruukku`) poikkeamat tästä. Jälleen saamme helposti laskettua näiden kahden käsittelyn luokkakeskiarvot. Miksei R sitten tulostanut kolmannen luokan kerrointa? Tämä johtuu siitä, että meillä on vain kolme `istutus`luokkaa, joten voimme arvioida vain kolme kerrointa. Viimeinen kerroin ei ole vapaasti estimoitava, sillä sen on oltava sellainen, että yleiskeskiarvoksi

[10] Ero §7.4 tuloksiin johtuu siitä, että nyt käytämme vain yhtä selittävää tekijää, joten vakio ei ole `emo 1` vaan "keskiemo".

tulee 52 cm. Esimerkissämme luokkien poikkeamien keskiarvosta on kumottava toisensa eli niiden summan on oltava 0. Viimeinen kerroin olisi siis $-3+4=1$.

Viimeinen valmis kontrasti (käyttäjä näet voi määritellä omia kontrastejaan) on *Helmert*:

```
> options(contrasts=c("contr.helmert","contr.poly"))
> summary(lm(pituus ~ istutus, data=taimet))

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  52.0000     0.7226  71.957  < 2e-16
istutus1      3.5000     0.8851   3.955  0.00191
istutus2     -0.5000     0.5110  -0.978  0.34715
```

Vakiotermi on yleiskeskiarvo, mutta kertoimet näyttävät käsittämättömiltä. Jostain salaperäisestä syystä S käyttää Helmertin kontrasteja oletusasetuksenaan, joten S-käyttäjän on syytä oppia muuttamaan kontrastit jommaksi kummaksi järkeväksi tyypiksi.

Kontrastit ovat vaihtoehtoisia tapoja esittää samat tulokset. Kukin niistä painottaa erilaisia aspekteja. Kaikki tuottavat samat sovitetut arvot (esim. funktiossa `predict.lm`). R käyttää oletuksenaan käsittelykontrasteja, ilmeisestikin siksi, että näin vakion ja kertoimen merkitys on sama kuin regressioanalyysissä joten on helppo käsitellä malleja, joissa on sekä jatkuvia muuttujia että luokkamuuttujia selittäjinä. Varsinkin suunnitelluissa kokeissa on usein jokin "nolla-taso" tai "kontrolli", joka on luontevaa valita "vakiotermiksi" — faktorin luokathan pystyi järjestämään komennon `factor` määritteellä `levels` (sivu 44). Summakontrasteja taas käytetään silloin kun yleiskeskiarvo ja poikkeamat siitä ovat kiinnostavia, kuten havaintoaineistojen analyysissä. Suunnitelluissa kokeissa käsittelyjen keskiarvo sen sijaan on harvoin mielekäs tai kiinnostava.

7.6 Varianssianalyysi: tulosten graafinen esitys

Komento `plot` tuottaa samanlaisen diagnostisen grafiikan kuin regressioanalyysi (kuva 6) riippumatta siitä, onko käytetty komento ollut `lm` vai `aov`. Männyntaimiesimerkissä diagnostiikka näyttää mainiolta ja voimme olla malliimme tyytyväisiä.

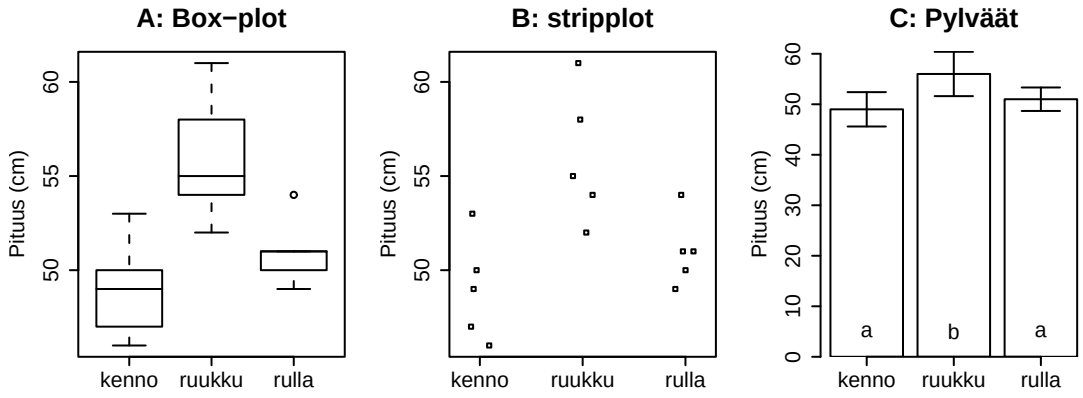
R käyttää luokitellun aineiston graafisen esityksen oletuskuvana rinnakkaisia `boxplot`-kuvioita (kuva 8A), mutta varsinkin pienissä aineistoissa niiden tunnusluvut, mediaani ja etenkin kvartiilit ovat epäluotettavia. Alkuperäiset havaintoarvot esittävä pistekuvio (`stripchart`) on tässäkin esimerkissä parempi vaihtoehto (kuva 8B).

```
>  par(mfrow=c(1,3))                        # 3 osakuvaa
>  plot(istutus, pituus, ylab="Pituus (cm)", main="A: Boxplot")
>  stripchart(pituus ~ istutus, vertical=T, method="jitter",
+  ylab="Pituus (cm)", main="B: stripchart")
```

Ekologiassa (kuten myös monilla biologian, lääketieteen ja käyttäytymistieteiden aloilla) on tapana esittää ANOVAn tulokset pylväsdiagrammina, jossa pylvään korkeus näyttää luokkakeskiarvon ja kuhunkin pylvääseen liitetään ns. virhejana. Ekologiassa on kaksi vahvaa koulukuntaa: toiset vaativat käytettäväksi virhejanana keskihajontaa, toiset taas keskiarvon keskivirhettä (pieni vähemmistö käyttää keskiarvon 95 % luottamusväliä). Kumpikaan käytäntö ei välttämättä ole hyvä. Kerrattakoon ensin eräitä perusasioita:

- Keskihajonta kuvaa yksittäisten *havaintojen* vaihtelua luokkakeskiarvon suhteen. Keskihajonta ei pienene otoskoon kasvaessa.
- Keskiarvon keskivirhe kuvaa *luokkakeskiarvon* tarkkuutta. Keskivirhe pienenee otoskoon kasvaessa ja keskiarvon tullessa luotettavammaksi.

Koska varianssianalyysissä vertaillaan nimenomaan keskiarvoja, keskiarvon keskivirhe näyttää pinnalta katsoen luontevalta tunnusluvulta. Ongelmana on kuitenkin se, että yksittäisten keskiarvojen keskivirheiden avulla ei voi kovinkaan hyvin hahmottaa luokkakeskiarvojen *erotusten*



Kuva 8: A: Boxplot on R:n oletuskuva luokitteluille. B: Pienelle aineistolle niitäkin parempi on stripchart. C: Ekologit käyttävät yleensä pylväsdiagrammia ja virhejanaa osoittamaan keskiarvoa ja virhettä (tässä 95 % luottamusväli) sekä merkitsevät samalla kirjaimella pylväät, jotka "eivät eroa merkitsevästi toisistaan". Aineisto `taimet`.

(tai yleisempien kontrastien) luottamusvälejä tai "merkitsevyyttä". Ongelma ei poistu vaikka käytettäisiinkin luokkakeskiarvojen 95% luottamusvälejä. Toisaalta keskihajonnan sisältävä pylväskuvio on sekin data-mustesuhteeltaan huono ja epäinformatiivisempi kuin `boxplot` ja `stripchart`.

Vaikka seuraavassa annammekin ohjeet tällaisten pylväskuvioiden piirtämiseksi, emme suosittele niiden käyttämistä. Todettakoon, että yhdessäkään varsinaisessa tilastotieteen oppikirjassa, jossa tavanomaisia tilastokuvioita käsitellään, ei tätä kuviotyyppiä esiinny. Päin vastoin, silloin kun tilastotieteilijät ovat niitä kommentoineet, on arvio ollut kielteinen.

Pylväsdiagrammien piirtäminen ja varsinkin virhejanojen lisääminen on hieman mutkallista (kuva 8B):

```
> ka <- tapply(pituus,istutus,mean)        # Luokkakeskiarvot
> hajonta <- tapply(pituus,istutus,sd)     # Luokkahajonnat
> n <- tapply(!is.na(pituus),istutus,sum)  # n luokittain
> se <- hajonta/sqrt(n)                    # Keskivirheet
> se <- tapply(pituus,istutus, function(x) sqrt(var(x)/length(x)))
> ci <- se*qt(0.975,n-1)                   # Luottamusrajat
> mp <- barplot(ka, ylim=c(0,max(ka+ci)), ylab="Pituus (cm)",
+ main="C: Pylväät", col=NULL)
> arrows(mp,ka-ci, mp,ka+ci, angle=90, code=3, length=0.1)
> text(mp, 5, c("a","b","a"))
```

Meidän on ensin laskettava luokkakeskiarvot sekä virhejanan pituus. Tällä kertaa käytämme 95 % luottamusrajoja, lähinnä näyttääksemme kuinka ne lasketaan. Keskihajontaa varten on valmis funktio, mutta keskivirhe on laskettava itse ja sitä varten tarvitsemme lukumäärät luokissa (`n`). Keskivirheen voi laskea myös ilman välivaiheita, käyttämällä nimetöntä funktiota `tapply`–komennossa. Hieman yllättävän näköinen tapa laskea havaintojen luvut luokittain (`n`) takaa, että tulos ottaa huomioon mahdolliset puuttuvat tiedot muuttujassa `pituus`. Puuttuvia tietoja ei tällä kertaa ole, joten ilmeisempi `table(istutus)` olisi tuottanut samat tulokset. Komento `qt(p,df)` antaa kriittisen t-arvon. Käytimme todennäköisyyttä $p = 0.975$, joka jättää häntään 2.5 %, ja näin kaksisuuntaisena antaa 95 % luottamusrajat.

Komento `barplot` tuottaa pylväsdiagrammin ja `ylim` takaa että y-akseli skaalataan niin, että virhejanatkin mahtuvat kuvaan. Yleensä R:n oletusgrafiikka on mustavalkoista, ja värit on lisättävä komennoilla, mutta `barplot` tuottaa värikuvia, jotka eivät sovi tähän mustavalkoiseen oppaa-

seen, joten olen poistanut värit (`col=NULL`). Komento `barplot` palauttaa pylväiden keskiviivojen x-koordinaatit (`mp`), joita käytämme komennossa `arrows` sijoittamaan virhejanat x-suunnassa. Kulma 90° määrittelee nuolenpään kulmaksi suoran, ja `code=3` kertoo, että nämä suorat "nuolenpäät" piirretään kumpaankin päähän janaa.

Ekologien yleensä käyttämät kirjaimet (`a, b,...`) tarkoittavat, että luokat ovat parittaisessa vertailussa merkitsevästi erilaiset, mikäli niissä ei ole yhteistä kirjainta. Merkitsevä ANOVA-tuloksemme näyttäisi aiheutuvan yhdestä ainokaisesta merkitsevästä erosta: `ruukku`taimet ovat pitempiä kuin muut. Tällainen *post hoc* vertailu on tavallinen käytäntö ANOVAn jälkeen ja useimmissa tilasto-ohjelmistoissa on laaja valikoima vertailumenetelmiä. Ei kuitenkaan R:ssä, missä voimme käyttää funktioita `pairwise.t.test` (§6.1.4, sivu 28) tai funktiota `TukeyHSD`. Erillisissä paketeissa on todennäköisesti lisää vaihtoehtoja, mutta R:n kehittäjät suhtautuvat monivertailuihin penseästi, koske niitä yleensä käytetään väärin ja ne ovat hyväksyttäviä vain harvinaisissa tapauksissa. ANOVA olettaa että ainoa ero luokkien välillä on keskiarvo ja kaikkien luokkien sisävaihtelu on sama. Erimittaisten hajontajanojen piirtäminen on vastoin ANOVAn oletusta ja näin ollen testimme ja kuvamme ovat ristiriidassa. Samaten monivertailut tehdään jälkikäteen ja ne eivät välttämättä ole yhteneväisiä ANOVAn kanssa: on mahdollista saada merkitsevä ANOVA ilman ainokaistakaan parittain merkitsevää eroa tai epämerkitsevä ANOVA ja useita parittain merkitseviä eroja.

7.7 Kiinteät ja satunnaiset vaikutukset

Varianssianalyysi on nimenomaan suunnitellun kokeen analyysin tarkoitettu menetelmä. Suunnitellulla kokeella tarkoitetaan, että tutkija on tietoisesti päättänyt käsittelyt ja niiden tasot. Päätettyjä tasoja sanotaan *kiinteiksi vaikutuksiksi* ("*fixed effects*"). Analyysissä on kuitenkin joskus mukana myös tekijöitä, joihin tutkijalla ei ole täyttä vaikutusvaltaa. Esimerkiksi kenttäkoe joudutaan ehkä tekemään niin, että valitaan joukko näytealoja, joiden sisällä käsittelyt tehdään. Nämä näytealat ovat vain umpimähkäinen (tai ihanteellisesti satunnainen) otos kaikista mahdollisista paikoista, joissa koe olisi voitu tehdä. Tutkija tietää vain, että paikat ovat luonnostaankin erilaisia, mutta hän ei pysty säätelemään tuota erilaisuutta. Tällaisia tekijöitä sanotaan *satunnaisiksi vaikutuksiksi* ("*random effects*").

Havainnoivassa tutkimuksessa kerätään joskus suuriakin aineistoja ilman ainokaistakaan kontrolloitua (eli kiinteää) tekijää, vaan kaikki tekijät ovat havaittuja. Tällaisiakin aineistoja analysoidaan ANOVAlla. Biometrian kirjan esimerkki satunnaisvaikutuksista (Ranta *et al.*, 1989, Esimerkki 8.5) on juuri tuollainen havainnoiva tutkimus, jossa on vain yksi satunnaisvaikutus (kaupunginosa). Tällaisen mallin analysointi ei ole erityisen kiinnostavaa, sillä siinä voi käyttää aivan tavallista lineaarista mallia (`lm` tai `aov`). Ainoa ero perus-ANOVAan on tulkinnassa, sillä tekijät eivät ole tutkijan hallitsemia. S-PLUS-ohjelmassa lienee erillinen komento (`raov`) tasapainoisille malleille, joissa on vain satunnaisia tekijöitä, mutta ainakin tätä kirjoitettaessa sellaista ei näkynyt olevan R:ssä, missä mallit on analysoitava samoilla menetelmillä kuin kiinteiden vaikutusten mallit.

Sangen usein tutkijalla on malli, jossa on sekä kiinteitä että satunnaisia vaikutuksia. Pääesimerkkimme (aineisto `taimet`) edustaa juuri tällaista tapausta: `istutus` on tutkijan päättämä kiinteä vaikutus, mutta käytetyt viisi `emo`puuta ovat vain umpimähkäinen otos kaikista mahdollisista emopuista. Muuttuja `emo` on siis satunnainen tekijä. Meidän pitäisi itse asiassa sovittaa kiinteiden ja satunnaisten vaikutusten sekamalli ("*mixed effects model*").

Yksinkertaisia sekamalleja pystyy sovittamaan funktiolla `aov`, jonka mallilauseeseen voi sisällyttää termin `Error`, joka kertoo minkä tason ("*stratum*") sisällä virhevaihtelu arvioidaan:

```
> taimet.mixed <- aov(pituus ~ istutus + Error(emo), data=taimet)
> summary(taimet.mixed)

Error: emo
          Df Sum Sq Mean Sq F value Pr(>F)
```

```
Residuals  4   54.0    13.5

Error: Within
          Df Sum Sq Mean Sq F value   Pr(>F)
istutus    2    130      65      13 0.003065
Residuals  8     40       5
> model.tables(taimet.mixed)
Tables of effects

 istutus
 kenno ruukku  rulla
    -3      4     -1
```

Kiinteän vaikutuksen `istutus` tunnusluvut ovat samat kuin kaksisuuntaisessa ANOVAssa aiemmin (sivu 45), kiitos yksinkertaisen ja tasapainoisen koeasetelman. Sen sijaan satunnaisvaikutus `emo` on tässä taulukossa vain kuvattu, ei lainkaan testattu. Mallitaulukko antaa niin ikään vain kiinteiden tekijöiden vaikutukset. Tämä on tyypillinen käytäntö sekamalleissa: satunnaistekijät ovat satunnaisia, joten emme ole lainkaan kiinnostuneet satunnaistekijän yksittäisen tason vaikutuksesta, ainoastaan niiden vaikutuksesta virheen suuruuteen. Toinen huomattava seikka on, että satunnaisvaikutuksilla ei oleteta olevan interaktioita (§7.8) kiinteiden vaikutusten kanssa. Ekologiselta kannalta tämä on tietystikin kestämätön oletus, mutta tehdään huomaamatta useimmiten kun sekamalleja sovitetaan.

Jos komennon `aov` mallilauseessa on `Error`-termi, tulos ei enää olekaan luokkien `lm` ja `aov` olio, joten sille ei esimerkiksi ole `plot`-komentoa.

Komennolla `aov` pystyy analysoimaan vain yksinkertaisia kiinteiden ja satunnaisten vaikutusten sekamalleja. Tutkijoilla on joskus huomattavan monimutkaisia ja hankalia malleja, joskus jopa tietoisen harkinnan tuloksena. SAS-ohjelmiston `MIXED`-proseduuri on suosittu tällaisten hankalien mallien analysointiin. Ainakin hyvin monet samanlaiset mallit pystyy analysoimaan myös R-paketilla `nlme` (Pinheiro & Bates, 2000). R:n verkkoarkistossa (`http://cran.r-project.org/`) on myös paketti `SASmixed`, jossa kerrotaan kuinka R:ssä pystyy analysoimaan monet SASsin *Mixed Models* -oppaan esimerkit. Katsokaamme siis, miltä äärettömän yksinkertainen, fiktiivinen esimerkkimme näyttää `nlme`-paketin ohjelmalla `lme` analysoituna (tulostus on jälleen hieman lyhennetty):

```
> library(nlme)
Loading required package: nls
> taimet.lme <- lme(pituus ~ istutus, data=taimet, random = ~ 1|emo)
> summary(taimet.lme)
Linear mixed-effects model fit by REML
 Data: taimet
      AIC      BIC    logLik
  72.1691 74.59363 -31.08455

Random effects:
 Formula: ~1 | emo
        (Intercept) Residual
StdDev:    1.683951 2.235821

Fixed effects: pituus ~ istutus
               Value Std.Error DF  t-value p-value
(Intercept)       49  1.251766  8 39.14471  <.0001
istutusruukku      7  1.414057  8  4.95029  0.0011
istutusrulla       2  1.414057  8  1.41437  0.1950

Number of Observations: 15
```

```
Number of Groups: 5
```

Komennolle `lme` annetaan mallilausekkeessa vain kiinteät vaikutukset (`istutus`). Satunnaiset tekijät annetaan erikseen määritteellä `random`. Satunnaistekijät määritellään yksipuolisella mallilausekkeella, jossa on vain riippuvuusoperaattori (`~`) ja oikea puoli. Tällä kertaa oikea puoli kertoo, että keskiarvo (`1`) on ehdollinen (`|`) satunnaistekijälle `emo`.

Myös tulostus on pitkällinen. Sovitus tehdään rajoitetulla suurimman uskottavuuden menetelmällä (REML), joten neliösumman sijaan päätunnusluku on log–uskottavuus (`logLik`). Log-uskottavuuden lisäksi annetaan kaksi muuta tunnuslukua: Akaiken informaatiokriteeri (AIC) sekä bayesilainen informaatiokriteeri (BIC). Sekä AIC että BIC ovat hierarkisten mallien vertailuun tarkoitettuja kriteerejä, jotka pohjautuvat log–uskottavuuten (logLik) ja estimoitujen kertoimien (parametrien) määrään p sekä BIC myös havaintojen lukumäärään n:

$$\text{AIC} = -2\text{logLik} + 2p \tag{7}$$

$$\text{BIC} = -2\text{logLik} + p\log(n) \tag{8}$$

Aina kun parametrien p määrää lisätään, -2logLik pienee, joten informaatiokriteerit rankaisevat ("penalisoivat") tästä lisäyksestä. Hierarkisessa mallin rakentamisessa valitaan usein se malli, jonka penalisoitu informaatiokriteeri (AIC tai BIC) on pienin.

Samoin kuin `aov`, myös `lme` antaa satunnaistekijästä vain kuvauksen ilman testejä. Tällä kertaa tulostetaan myös "varianssin komponentit" eli satunnaistekijästä johtuva vaihtelu (`Intercept`) sekä satunnaistekijän poistamisen jälkeen jäävä sisävaihtelu (`Residual`). Mallikertoimet tulostetaan samalla tavoin kuin funktiossa `lm`.

Funktio `lme` palauttaa luokan `lme` olion, joka *ei* ole perinyt ominaisuuksia luokalta `lm`. Näin ollen useimmat luokan `lm` menetelmät eivät sovi, mutta luokalla `lme` on joitain omia menetelmiä (katso `help(lme)`). Menetelmää `anova` voi kuitenkin käyttää hierarkisten `lme`-mallien vertailuun.

Funktio `lme` sekä muut paketin `nlme` funktiot ovat erittäin voimakkaita ja monipuolisia. Niitä voi käyttää myös esim. toistuvien mittausten, aikasarjojen ja spatiaalisten transektien analyysiin. Tämä kuitenkin ylittää tälle oppaalle asetetut puitteet. Ohjelmien mukana tuleva dokumentaatio (esim. `SASmixed` -tekstit) sekä Venables & Ripley (1999) ovat hyvä lähtökohta opiskeluun. Pinheiro & Bates (2000) johdattavat syvemmälle.

7.8 Interaktiot eli yhdysvaikutukset

Tilastollinen interaktio tarkoittaa, että emme voi tarkastella pelkästään tekijöiden *päävaikutuksia* erillisinä, vaan meidän on otettava huomioon myös muiden tekijöiden vaikutus yhtä tekijää tarkastellessamme. Toinen tekijä voi joko voimistaa tai heikentää päävaikutusta.

Biometrian kirjassa (Esimerkki 8.16 Ranta *et al.*, 1989, Aineisto `kammio` paketissa `rekola`) tarkastellaan otsonin ja typen oksidien vaikutusta kuusen kloroplastin pituuteen. Koe on tehty kaasukammiossa, jossa NOX–tasot olivat 0 tai 5 μg m^{-3} ja O_3–tasot 0 tai 40 μg m^{-3} (kirjassa yksikkövirhe). Käsittelyjä oli kaikkiaan neljä: ei lainkaan lisättyjä kaasuja ("kontrolli"), lisätty NOX, lisätty otsoni ja lisätty sekä NOX että otsoni.

Interaktiomalli voidaan määrittää kahdella vaihtoehtoisella tavalla malliyhtälössä:

```
lm(NOX + O3 + NOX:O3)
    lm(NOX*O3)
```

Ensimmäinen mallilauseke luettelee julkisesti sekä päävaikutukset (`NOX`, `O3`) että niiden interaktion (`NOX:O3`). R käyttää interaktion operaattorina merkkiä "`:`" — monissa muissa ohjelmisssa operaattori on "`.`", joka kuitenkin on R:ssä sallittu muuttujien nimissä.

Jälleen mallin määrittely on yhtä helppoa kuin mallilausekkeen kirjoittaminen:

```
> data(kammio)
> kammio.aov <- aov(kloropl ~ NOX*O3, data=kammio)
> summary(kammio.aov)
            Df   Sum Sq  Mean Sq F value    Pr(>F)
NOX          1 0.218700 0.218700 59.6455 5.622e-05
O3           1 0.020833 0.020833  5.6818 0.0442922
NOX:O3       1 0.172800 0.172800 47.1273 0.0001290
Residuals    8 0.029333 0.003667
```

Käytimme tällä kertaa funktiota `aov`, mutta saisimme funktion `lm` mallisen tulostuksen pyytämällä yhteenvetoa komennolla `summary.lm`.

Interaktion vaikutus on merkitsevä, eli typen vaikutus on erilainen otsonin kanssa ja ilman otsonia. Nämä erot näkyvät vielä paremmin vaikutustaulukosta:

```
> model.tables(kammio.aov)
Tables of effects

 NOX
     0      5
 0.135 -0.135

 O3
       0       40
-0.04167  0.04167

 NOX:O3
   O3
NOX 0     40
  0 -0.12  0.12
  5  0.12 -0.12
```

Tämän taulukon avulla voimme arvioida kunkin käsittely-yhdistelmän kokonaisvaikutukset, jotka esitämme samassa järjestyksessä kuin yhdysvaikutukset mallitaulukossa, eli riveillä typen oksidien ja sarakkeilla otsonin vaikutukset:

$$\begin{bmatrix} +0.135 - 0.042 - 0.120 & +0.135 + 0.042 + 0.120 \\ -0.135 - 0.042 + 0.120 & -0.135 + 0.042 - 0.120 \end{bmatrix} = \begin{bmatrix} -0.027 & +0.297 \\ -0.057 & -0.213 \end{bmatrix}$$

Jos yhdysvaikutuksia ei olisi, saisimme ennustetun kloroplastipituuden laskemalla yhteen pelkästään päävaikutukset. Merkitsevä yhdysvaikutus tarkoittaa, että kullakin käsittely-yhdistelmällä on oma vaikutuksensa, joka on lisättävä päävaikutuksiin. Tällä kertaa tilanne on sangen selvä: Yhdysvaikutus likipitäen kumoaa päävaikutukset ensimmäisessä sarakkeessa ja vahvistaa niitä toisessa sarakkeessa. Toisin sanoen, typenoksideilla ei ole juuri lainkaan vaikutusta kloroplasteihin ilman otsonia, mutta otsonipitoisuudessa 40 μg m^{-3} typen oksidien vaikutus on voimakas. Otsoni voimistaa typenoksidien vaikutusta eli vaikutuksen sanotaan olevan "synergistinen" (vastakohta olisi "antagonistinen"). Vaikutustaulukko käyttää "summakontrasteja" (§7.5), mikä ei ole erityisen mielekästä tällaisessa suunnitellussa kokeessa, missä olisi luontevampaa verrata käsittelyjä nollatasoihin.

Varianssitaulukko antaa merkitsevyydet sekä päävaikutuksille että yhdysvaikutuksille. Tällä kertaa kaikki ovat merkitseviä, vaikkakin `O3` hipoo sovinnaisia rajoja. Jos interaktio on merkitsevä, päävaikutusten merkitsevyyteen ei kuitenkaan voi kiinnittää paljonkaan huomiota. Vaikka otsonin päävaikutus olisi epämerkitsevä, otsoni olisi kuitenkin tärkeä tekijä, sillä se on osallinen merkitsevään interaktioon. Kaikkia merkitsevään interaktioon sisältyviä termejä on pidettävä merkitsevinä riippumatta niiden päävaikutusten analyysistä.

Yhdysvaikutukset ovat tärkeitä ekologisessa analyysissä. Syynä siihen ettemme analysoineet niitä männyntaimiesimerkissä (§7.4) oli aineiston pieni koko, mikä ei sallinnut interaktiotermien

analyysiä. Olisi ollut mahdollista että joidenkin emopuiden jälkeläiset sopivat paremmin esim. kennotaimiksi ja tällöin olisimme saanet merkitsevän emo:istutus -interaktion. Meillä oli kuitenkin vain yksi toisto kustakin käsittelykombinaatiosta, joten emme pystyneet arvioimaan kombinaatioiden sisävaihtelua emmekä siis myöskään yhdysvaikutusten merkitsevyyttä. Ekologisia perusteita meillä ei ollut yhdysvaikutusten laiminlyöntiin, ainoastaan teknisiä.

Kiinteiden ja satunnaisten vaikutusten sekamalleissa (§7.7) on myös tapana olettaa että satunnaistekijöillä ei ole interaktioita keskenään eikä kiinteiden tekijöiden kanssa. Tämä on usein perusteeton olettamus, mutta se on tapana tehdä. Perusteet ovat jälleen käytännölliset ja tekniset, eivät tieteelliset. Mikäli meillä on havaintoaineisto, satunnaisten tekijöiden yhdysvaikutusten analysointi on usein tieteellisesti välttämätöntä.

Mikäli meillä on monitekijäinen malli, yhdystermien lukumäärä kasvaa usein hyvin nopeasti. Kahden tekijän mallissa meillä oli vain ensimmäisen asteen interaktiot, mutta jo kolmen tekijän malli laajenee hankalan suureksi:

$$\texttt{lm(A*B*C)} = \texttt{lm(A + B + C + A:B + A:C + B:C + A:B:C)}$$

Neljän tekijän ajattelemisenkin pitäisi herättää väristyksiä. Korkean asteen yhdysvaikutukset ovat usein hankalia tulkita, ja lisäksi suurten mallien estimointi on epävarmaa. Tämän takia korkeimman asteen interaktiot jätetään usein pois malleista. Tämä voidaan kätevästi lyhentää mallilausekkeessa:

$$\texttt{lm(A + B + C)\^{}2} = \texttt{lm(A + B + C + A:B + A:C + B:C)}$$

7.9 Järjestetyt faktorit

Ranta *et al.* (Esimerkki 8.12 1989, aineisto pojoviken paketissa rekola) käsittelevät neljältä eri paikalta (muuttuja paikka) Pohjanpitäjänlahdelta kerättyä aineistoa, jossa on määritetty eläinplanktonin biomassat kuudelta eri syvyydeltä (muuttuja syvyys, arvot 0.2, 0.5, 1, 2, 5 ja 10 m).

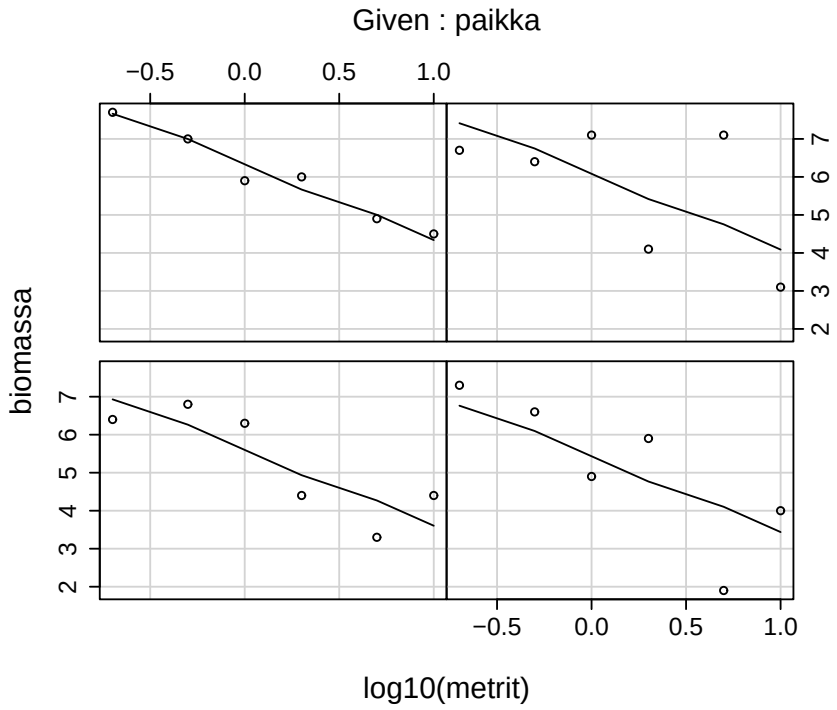
Kumpaakin tekijää käsitellään faktoreina ja suoritetaan kaksisuuntainen varianssianalyysi. Koska syvyyksiä ei ole replikoitu paikoittain, emme pysty analysoimaan interaktioita. Meillä ei ole mitään tieteellistä perustetta tähän laiminlyöntiin, sillä emme voi olettaa syvyyden vaikutuksen olevan paikasta riippumaton. Perusteemme on vain aineiston vähäisyys. ANOVA-malli on yksinkertainen:

```
> pojo.aov <- aov(biomassa ~ syvyys + paikka, data=pojoviken)
> summary(pojo.aov)
            Df Sum Sq Mean Sq F value   Pr(>F)
syvyys       5 31.652   6.330  4.8161 0.007964
paikka       3  3.141   1.047  0.7966 0.514715
Residuals   15 19.716   1.314
```

Paikkojen välillä ei ole merkitsevää eroa, mutta syvyyksien välillä on.

Perinteinen ANOVA-taulukko on tutkijan kannalta hieman ongelmallinen, sillä se ei ota huomioon, että syvyydet ovat järjestettyjä: $0.2 < 0.5 < 1 < 2 < 5 < 10$. Merkitsevä testitulos osoittaa vain, että *jokin* syvyys poikkeaa muista. Tutkijan kannalta kuitenkin on kiinnostavaa tietää, onko mitään järkeä siinä *mikä* pituus poikkeaa muista. Onko biomassa korkeimmillaan pinnan lähellä ja laskee alaspäin? Vai päin vastoin? Vai onko biomassan maksimi välivedessä? Vai sahaako biomassa tolkuttomasti edestakaisin? Tutkija pitää syvyyttää järjestettynä, mutta perus-ANOVA pitää sitä järjestämättömänä. Ei kuitenkaan R, joka tuntee järjestetyt faktorit:

```
> summary.lm(pojo.aov)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.26667    0.46805  11.252 1.04e-08
```



Kuva 9: Eläinplanktonin biomassat eri syvyyksillä. Kukin osakuva esittää yhtä tutkimuspaikkaa

```
syvyys.L    -2.78189    0.57324  -4.853 0.000211
syvyys.Q    -0.05183    0.57324  -0.090 0.929157
syvyys.C     0.40808    0.57324   0.712 0.487460
syvyys^4     0.06142    0.57324   0.107 0.916095
syvyys^5    -0.03307    0.57324  -0.058 0.954755
paikka2     -0.16667    0.66192  -0.252 0.804619
paikka3      0.73333    0.66192   1.108 0.285360
paikka4      0.48333    0.66192   0.730 0.476519
```

Järjestetyn faktorin voi luoda tai järjestämättömän faktorin muuttaa järjestetyksi komennoilla:

```
>  syvyys <- factor(syvyys, ordered=T)
>  syvyys <- ordered(syvyys)  ## Jos syvyys oli jo faktori
```

Tarpeen vaatiessa voidaan määritteellä `levels` luetella tasojen järjestys.

Järjestetyille faktoreille käytetään *polynomisia kontrasteja* (vrt. §7.5). Niissä luokan vaikutus jaetaan lineaariseen komponenttiin (`L`), kvadraattiseen komponenttiin (`Q`), kolmannen asteen komponenttiin (`C`) sekä korkeamman asteisin komponentteihin. Tämän on vain tapa jakaa sama vaikutus eri termeille, kuten muissakin kontrasteissa. Lopulliset ennustetut arvot ovat samat kuin järjestämättömillä faktoreilla, samaten ANOVA-taulukko. Jotta polynomiset kontrastit olisivat kelvollisia, faktorin on oltava jotakuinkin tasavälinen. Näin onkin tällä kertaa, sillä log(syvyys) on liki tasavälinen (kuva 9).

Ilmeisesti log–syvyyden vaikutus on liki lineaarinen, sillä ylemmän asteiset termit ovat melko merkityksettömiä. Voimme vielä testatakin tämän:

```
> m <- as.numeric(syvyys)  ## Tasot numeroiksi 1 ... 6
> pojo.L <- lm(biomassa ~ poly(m,1) + paikka, data=pojoviken)
> anova(pojo.L, pojo.aov)
Analysis of Variance Table
```

```
Model 1: biomassa ~ poly(m, 1) + paikka
Model 2: biomassa ~ syvyys + paikka
  Res.Df Res.Sum Sq Df  Sum Sq F value Pr(>F)
1     19    20.4126
2     15    19.7162  4  0.6963  0.1324  0.968
```

Funktio `poly(x,n)` muodostaa `n` asteen polynomin vektorista `x`, ja käytimme mallilausekkeessamme vain ensimmäisen asteen polynomia eli lineaarista termiä. Mallien `pojo.L` ja `pojo.aov` ero on, että edellinen sisältää vain lineaarisen vaikutuksen, jälkimmäinen myös ylemmän asteiset vaikutukset aina viidenteen asteeseen. ANOVAmme osoittaa, että ylemmän asteiset polynomitermit eivät paranna oleellisesti malliamme, joten voimme pitää biomassan muutosta likipitäen lineaarisena suhteessa log–syvyyteen.

Nyt pystymme myös korjaamaan laiminlyöntimme interaktion tutkimisessa: Voimme testata, onko syvyyden vaikutus merkitsevästi erilaista eri paikoissa. Alkujaan vapausasteet eivät riittäneet tähän, mutta kun tutkimme vain syvyyden lineaarista komponenttia, voimme analyysin tehdä:

```
> pojo.Lint <- aov(biomassa ~ poly(m,1)*paikka, data=pojoviken)
> summary(pojo.Lint)
                  Df  Sum Sq Mean Sq F value    Pr(>F)
poly(m, 1)         1 30.9558 30.9558 25.3473 0.0001221
paikka             3  3.1412  1.0471  0.8574 0.4831290
poly(m, 1):paikka  3  0.8724  0.2908  0.2381 0.8684761
Residuals         16 19.5402  1.2213
```

Interaktioita ei tarvita.

Lopuksi katsomme kuvaa. `coplot`–kuvaan voi kirjoittaa oman paneelifunktion, joka korvaa oletusvaihtoehdon esittää vain havaintopisteet. Paneelifunktion oletustoiminto on:

```
panel=function(x, y, ...) {points(x,y)}
```

Funktiolle välitetään kyseiseen paneeliin tulevien havaintopisteiden arvot ja oletuksena on lisätä ne kuvaan. Me haluamme mukaan myös mallin `pojo.L` sovitetut arvot, ja sen takia paneelifunktiolle on välitettävä myös tieto paneeliin kuuluvista havainnoista määritteellä `subscripts`. Kuva 9 syntyy komennoilla (sulkujen kanssa kannattaa olla tarkkana):

```
> metrit <- as.numeric(levels(syvyys)[syvyys]) # Tasot numeroiksi 0.2 ... 10
> coplot(biomassa ~ log10(metrit) | paikka, show.given=F, subscripts=T,
+    panel=function(x,y,subscripts,...) { points(x,y)
+    lines(x,fitted(pojo.L)[subscripts]) })
```

Esimerkistä näkyy myös, kuinka faktorin voi muuttaa numeeriseksi muuttujaksi. Sovitetut arvot ovat melkein suorat: lievä aaltoilu johtuu siitä, että järjestetty faktori ei ollut aivan tasavälinen.

7.10 "Kovarianssianalyysi"

"Kovarianssianalyysi" on vanhentunut termi. Se on peräisin ajalta ennen lineaarisia malleja, jolloin jatkuvien muuttujien ja luokkamuuttujien yhdistäminen nähtiin merkillisenä ja oman terminsä arvoisena. Lineaarisissa malleissa kovarianssianalyysi syntyy yksinkertaisena malliyhtälönä. Muinoin kovarianssianalyysiä käytettiin silloin, kun suunnitellun kokeen kiinteiden tekijöiden lisäksi oli joitain satunnaisia jatkuvia muuttujia. Päähuomio kohdistui kiinteisiin luokkamuuttujiin, mutta jatkuvien "taustamuuttujien" vaikutus haluttiin poistaa ennen varsinaiseen varianssianalyysiin ryhtymistä. Nykykielellä voisimme sanoa, että kovarianssianalyysi on satunnaisten ja kiinteiden vaikutusten sekamalli, jossa jatkuvat muuttujat ovat satunnaistekijöitä. Näinkin määriteltynä mallin sovittaminen poikkeaa vanhoista ajoista, sillä nykyään kaikki tekijät sovitetaan yhtaikaa eikä peräjälkeen kuten muinoin.

Ranta *et al.* (Esimerkki 8.22 1989, aineisto `daphnia` paketissa `rekola`) käsittelevät perinteisen kovarianssianalyysin malliin aineistoa, jossa tutkitaan lämpötilan (muuttuja `lampo`, tasot 15, 20

ja 25°C) vaikutusta vesikirpun *Daphnia longispina* jälkeläistuottoon. Lisääntymiskertojen lukumäärä vaihteli vesikirpuittain, ja mitä ilmeisimmin tämä vaikuttaa jälkeläistuottoon. Lisääntymiskerrat (muuttuja `kerrat`) on siis satunnainen jatkuva kovariaatti. Graafinen tarkastelu tukee kuvaustamme (kuva 10).

Analysoimme aluksi täydellistä mallia, jossa on myös yhdysvaikutus lämpötilan ja lisääntymiskertojen välillä. Merkitsevä yhdysvaikutus viittaisi siihen, että eri lämmöissä syntyy kertaa kohti eri määrä jälkeläisiä:

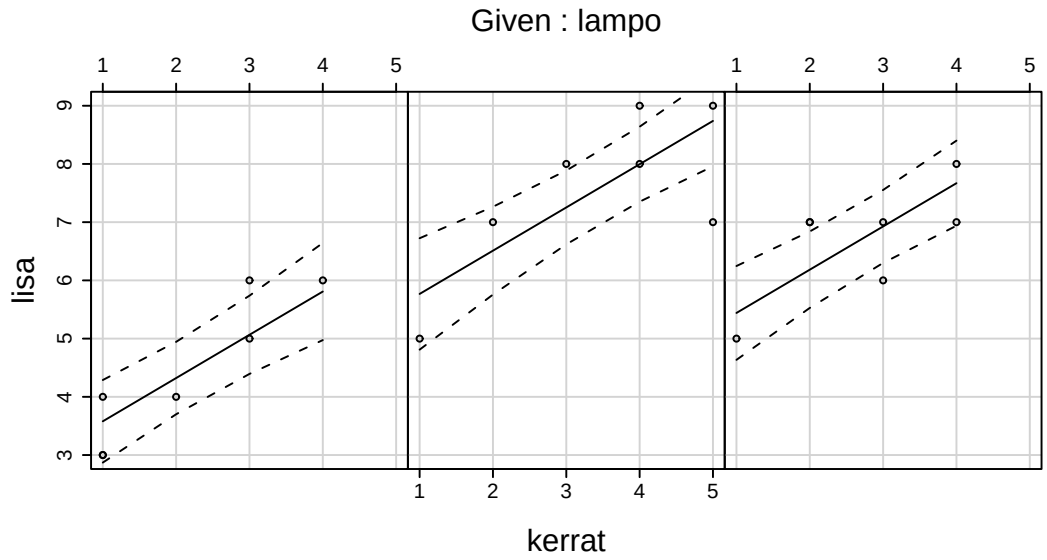
```
> daphnia.full <- lm(lisa ~ lampo*kerrat, data=daphnia)
> summary(daphnia.full)

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     4.23366    0.44670   9.478 1.01e-07
lampo.L         1.93840    0.75946   2.552 0.022095
lampo.Q        -1.24475    0.78770  -1.580 0.134907
kerrat          0.74699    0.15123   4.940 0.000178
lampo.L.kerrat -0.26275    0.28195  -0.932 0.366135
lampo.Q.kerrat  0.08562    0.24025   0.356 0.726520
```

Muuttuja `lampo` on luonnollisestikin käsitelty järjestettynä faktorina (§7.9), joten se on esitetty lineaarisena ja kvadraattisena kontrastina. Järjestämättömän faktorin oletusesityshän olisi ollut esittää lämpötilojen 20 ja 25°C erot tasoon 15°C. Interaktiotermit eivät näytä olevan merkitseviä, joten kokeilemme niiden poistamista ja analysoimme vain päävaikutuksia:

```
> daphnia.main <- lm(lisa ~ lampo + kerrat, data=daphnia)
> anova(daphnia.main, daphnia.full)
Analysis of Variance Table

Model 1: lisa ~ lampo + kerrat
Model 2: lisa ~ lampo + kerrat + lampo:kerrat
  Res.Df Res.Sum Sq Df  Sum Sq F value Pr(>F)
1     17    10.3020
2     15     9.6349  2  0.6671  0.5193 0.6052
> summary(daphnia.main)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   4.1864     0.4277   9.788 2.12e-08
lampo.L       1.3161     0.2998   4.390 0.000400
lampo.Q      -1.0265     0.3163  -3.245 0.004760
kerrat        0.7429     0.1421   5.227 6.84e-05
```

Päävaikutusmallin mukaan lämpötila on merkitsevä. Lämpötilan vaikutus näyttää olevan merkitsevästi kvadraattinen. Koska kvadraattisen termin kerroin on negatiivinen, malli ennustaa että vesikirpulla on optimi jossain lämpötila-akselilla. Epämerkitsevä kvadraattinen termi olisi merkinnyt, että vaste on lineaarinen. Kvadraattisuus näkyy myös kuvasta 10, missä jälkeläismäärät ovat suurimmat keskimmäisessä paneelissa. ANOVAn mukaan tämä ero on myös merkitsevä. Myös perinteisen ANOVAn malliin analysoituna, lämpötila on merkitsevä tekijä (saman tuloksen olisi tietysti saanut funktiolla `aov`):

```
> anova(update(daphnia.main, . ~ . -lampo), daphnia.main)
Analysis of Variance Table

Model 1: lisa ~ kerrat
Model 2: lisa ~ lampo + kerrat
  Res.Df Res.Sum Sq Df Sum Sq F value    Pr(>F)
```



Kuva 10: *Daphnia longispina*n jälkeläistuotto suhteessa lisääntymiskertoihin 15, 20 ja 25°C lämpötiloissa: havainnot sekä sovitetut arvot ja niiden 95 % luottamusvälit.

```
1     19     27.234
2     17     10.302  2 16.932  13.970 0.0002579
```

Lopuksi piirrämme kuvan, johon lisäämme havaintopisteiden lisäksi myös ennustetut arvot sekä niiden luottamysvälit. Suoria varten meidän on järjestettävä x–akseli (`sort.list`).

```
> daphnia.ci <- predict(daphnia.main, interval="confidence")
> coplot(lisa ~ kerrat | lampo, columns=3, show.given=F, subscripts=T,
+   panel=function(x,y,subscripts, ...) { points(x,y)
+   i <- sort.list(x); fv <- daphnia.ci[subscripts,]
+   matlines(x[i],fv[i,], col=1, lty=c(1,2,2)) } )
```

Aineiston olisi tietysti pystynyt analysoimaan myös siten, että muuttuja `kerrat` olisi määritelty satunnaistekijäksi funktioissa `aov` tai `lme` (paketti `nlme`, §7.7).

7.11 Epätasapainoiset mallit

ANOVAn perusesimerkkimme (`taimet`) on tasapainoinen ja täydellinen. Tämä tarkoittaa, että meillä on kaikkia käsittelykombinaatiot ja kaikista yhtä monta toistoa (tosin vain yksi). Perinteinen ANOVA on kehitetty juuri tällaisille tasapainoisille ja täydellisille asetelluille kokeille ("*designed experiments*"). Koska asetelmatekijät (käsittelyt) ovat teknisesti toisistaan riippumattomia, voimme analysoida niiden vaikutusta yhtaikaa yhdessä taulukossa.

Epätasapainoinen asetelma syntyy, kun meillä ei ole kaikkia käsittelykombinaatioita tai niissä on eri määrä toistoja. Joskus tutkija laatii jopa tarkoituksella tällaisen epätasapainoisen asetelman. Mikäli hän silloin joutuu ongelmiin analyysin kanssa, hän saa ansionsa mukaan. Sangen usein alkujaan tasapainoinen asetelma muuttuu epätasapainoiseksi tapaturmaisesti. Voimme kuvitella, että fiktiivisessä männynistutuskokeessa fiktiivinen hirvi (*Alces alces*) syö ensimmäisen taimen (`istutus kenno`, `emo 1`). Tällöin ANOVA-taulukot näyttävät tältä:

```
> attach(taimet)
> ### Negatiivinen indeksi poistaa havainnon
> summary(aov(pituus[-1] ~ istutus[-1] + emo[-1]))
            Df  Sum Sq Mean Sq F value  Pr(>F)
istutus[-1]  2 102.679  51.339  8.9844 0.01167
```

```
emo[-1]      4  42.750  10.687  1.8703 0.22052
Residuals    7  40.000   5.714
> summary(aov(pituus[-1] ~ emo[-1] + istutus[-1]))
            Df  Sum Sq Mean Sq F value   Pr(>F)
emo[-1]      4  28.929   7.232  1.2656 0.367551
istutus[-1]  2 116.500  58.250 10.1937 0.008441
Residuals    7  40.000   5.714
```

Käsittelyjen keskineliösummat, F-arvot ja P-arvot *muuttuvat* kun tekijöitä verrataan eri järjestyksessä. Tasapainoisessa ANOVAssa tekijöiden vertausjärjestys ei vaikuttanut merkitsevyyksiin (kokeile!), mutta yhden ainokaisen kuvitellun taimen tuho aiheutti huomauttavia seurauksia taulukkoomme. Onko siis käsittelyn `istutus` $p = 0.01167$ vai $p = 0.008441$? Hupaisaa kyllä, nämä arvot saisivat perinteisillä rajoilla eri määrän tähtiä, mikä on jälleen argumentti tähtien käyttöä vastaan.

Samat erot näkyvät myös mallitaulukoissa, joka mainitsee nyt myös toistojen määrät luokissa (`rep`):

```
> model.tables(aov(pituus[-1] ~ istutus[-1] + emo[-1]))
Tables of effects

 istutus[-1]
     kenno ruukku  rulla
    -2.679  3.571 -1.429
rep  4.000  5.000  5.000

 emo[-1]
     1     2     3    4    5
    -3 -1.25 -0.25 1.75 1.75
rep  2  3.00  3.00 3.00 3.00
> model.tables(aov(pituus[-1] ~ emo[-1] + istutus[-1]))
Tables of effects

 emo[-1]
         1      2       3     4     5
    -1.929 -1.429 -0.4286 1.571 1.571
rep  2.000  3.000  3.0000 3.000 3.000

 istutus[-1]
    kenno ruukku rulla
       -3    3.7  -1.3
rep     4    5.0   5.0
```

Istutuskäsittelyjen vaikutukset riippuvat siitä, analysoidaanko ne ennen vai jälkeen emojen vaikutuksen. Koska meillä on eri määrät toistoja käsittelyissä, vaikutusten suora summa ei olekaan enää nolla, mutta toistojen määrällä painotettu summa on. Riippuvuus on teknistä ja välttämätöntä. Tasapainoisessa asetelmassa kaikki `istutus`-vaikutukset arvioitiin samoista `emo`puista. Nyt sen sijaan käsittelyn `kenno` vaikutukset on arvioitu eri emopuista kuin muiden käsittelyjen vaikutukset.

Jos malli on epätasapainoinen, siitä ei voi laatia yhtä yksiselitteistä ANOVA-taulukkoa, joka osoittaisi tekijän merkityksen, sillä vertausjärjestys vaikuttaa tunnuslukuihin. Tämä on kiusallista. Muuan ratkaisu on laatia yksi yhteinen taulukko vaikka sellaista ei voi laatia. Tätä sovelletaan SAS-ohjelmassa sekä muissa ohjelmissa, jotka seuraavat sen esimerkkiä. SAS nimittää tällaisia mahdottomia neliösummia *tyypin III neliösummiksi*. "Tyypin III" neliösummat arvioidaan siten, että kukin tekijä *erikseen* poistetaan mallista, jossa kaikki muut tekijät ovat mukana ja nämä erikseen poistot raportoidaan *yhtaikaa* kaikille tekijöille. Toisin sanoen, tekijöiden vertailu perustuu eri malleihin. Tällä tavoin saadaan kätkettyä, ettei tekijöitä voi vertailla toisistaan riippumatta

epätasapainoisessa mallissa ja esitetään kuitenkin tämä mahdoton vertailu.

Tällainen syntinen vertailu on mahdollista tehdä myös R:ssä, mikäli sitä jostain syystä todella haluaa:

```
### Älä tee tätä kotona!
> drop1(aov(pituus[-1] ~ emo[-1] + istutus[-1]), test="F")
Single term deletions

Model:
pituus[-1] ~ emo[-1] + istutus[-1]
            Df Sum of Sq     RSS     AIC F value    Pr(F)
<none>                    40.000  28.698
emo[-1]      4    42.750  82.750  30.875  1.8703 0.220518
istutus[-1]  2   116.500 156.500  43.796 10.1937 0.008441
```

Tätä kannattaa verrata eri järjestyksissä sovitettuihin malleihin yllä. Mielestäni ainoa pätevä syy "tyypin III" neliösummien käyttöön on, ettei ymmärrä mitä tekee.

7.12 Mallin rakentaminen

Perinteisessä regressioanalyysissä mallin rakentaminen tapahtui aivan toisin kuin ANOVAssa. ANOVA oli suunnitellun koeasetelman analyysi, eli kaikki tekijät voitiin ja haluttiin analysoida. Regressionalyysiä sen sijaan käytettiin ennen kaikkea monimuuttujaisen havaintoaineiston analyysiin. Regressioanalyysin selittävät muuttujat olivat ANOVA-termein epätasapainoisia ja useimmiten satunnaistekijöitä. ANOVAssa oli myös tapana näyttää jokaisen koeasetelman tekijän merkitsevyys, myös silloin kun tekijällä ei ollut merkitsevää vaikutusta. Regressioanalyysiin sen sijaan haluttiin sisällyttää vain merkitseviä tekijöitä. Näin ollen regressioanalyysissä rakennettiin mallit siten, että kaikki mukana olevat tekijät olivat merkitseviä.

R:ssä on mallin rakentamista helpottamaan jo luvussa 7.11 nähty funktio `drop1` ("pudota yksi") sekä vastakkaisesti toimiva `add1` ("lisää yksi"). Funktio `drop1` kokeilee, kuinka mallin hyvyys muuttuisi, jos tekijä poistettaisiin. Tämän jälkeen tekijä voidaan poistaa ja tutkia pelkistettyä mallia. Ilman lisämääreitä kumpikaan funktio ei tulosta muita vertailukriteerejä kuin AIC:n (yhtälö 7 sivulla 52), mutta pyydämme myös F-tunnusluvun. Katsokaamme aluksi kuinka aineiston `plutakko` malli yksinkertaistuu:

```
> attach(plutakko)
> plut.cur <- lm(Fotosynt ~ Pikopl + Plankt + NH4 + PO4)
> drop1(plut.cur, test="F")
Single term deletions

Model:
Fotosynt ~ Pikopl + Plankt + NH4 + PO4
       Df Sum of Sq     RSS     AIC F value     Pr(F)
<none>               8.0973 -8.0841
Pikopl  1    3.4485 11.5458 -2.9882  6.3883 0.0232089
Plankt  1   11.8755 19.9728  7.9728 21.9991 0.0002902
NH4     1    0.0606  8.1579 -9.9350  0.1122 0.7422573
PO4     1    0.3043  8.4016 -9.3462  0.5638 0.4643633
> plut.cur <- update(plut.cur, . ~ . -NH4)  ## Poista NH4
> drop1(plut.cur, test="F")
Single term deletions

Model:
Fotosynt ~ Pikopl + Plankt + PO4
       Df Sum of Sq      RSS      AIC F value     Pr(F)
```

```
<none>                8.1579  -9.9350
Pikopl  1    3.9250  12.0828  -4.0789  7.6980 0.0135335
Plankt  1   12.0106  20.1684   6.1677 23.5563 0.0001761
PO4     1    0.2696   8.4275 -11.2847  0.5288 0.4776216
> plut.cur <- update(plut.cur, . ~ . -PO4)  ## Poista PO4
> drop1(plut.cur, test="F")
Single term deletions

Model:
Fotosynt ~ Pikopl + Plankt
       Df Sum of Sq      RSS      AIC F value    Pr(F)
<none>                8.4275 -11.2847
Pikopl  1    4.0185  12.4460  -5.4867  8.1062  0.01114
Plankt  1   13.6820  22.1095   6.0055 27.5994 6.47e-05
```

Komennolla `drop1` näimme aina huonoimman mallitermin, poistimme sen (`update`) ja jatkoimme kunnes kaikki jäljellä olevat termit olivat merkitseviä.

Komento `add1` toimii samoin, mutta tutkii termien lisäyksen vaikutusta, minkä jälkeen voimme aina lisätä kullakin askelella parhaan termin. Lisäyskomento `add1` tarvitsee myös tiedon, mitä termejä malliin voi harkita lisättäväksi ("*scope*"), eli suurimman harkittavan mallin yksipuolinen mallilause:

```
> plut.cur <- lm(Fotosynt ~ 1)
> add1(plut.cur, ~ Plankt+Pikopl+PO4+NH4, test="F")
Single term additions

Model:
Fotosynt ~ 1
       Df Sum of Sq     RSS     AIC F value     Pr(F)
<none>              27.0129  8.0116
Plankt  1   14.5669 12.4460 -5.4867 21.0674 0.0002272
Pikopl  1    4.9034 22.1095  6.0055  3.9920 0.0610550
PO4     1    1.8888 25.1241  8.5619  1.3532 0.2599121
NH4     1    0.1573 26.8556  9.8949  0.1054 0.7491703
> plut.cur <- update(plut.cur, . ~ . +Plankt) ## Lisää Plankt
> add1(plut.cur, ~ Plankt+Pikopl+PO4+NH4, test="F")
Single term additions

Model:
Fotosynt ~ Plankt
       Df Sum of Sq      RSS      AIC F value   Pr(F)
<none>               12.4460  -5.4867
Pikopl  1    4.0185   8.4275 -11.2847  8.1062 0.01114
PO4     1    0.3632  12.0828  -4.0789  0.5110 0.48443
NH4     1    0.6712  11.7748  -4.5954  0.9690 0.33873
> plut.cur <- update(plut.cur, . ~ . +Pikopl) ## Lisää Pikopl
> add1(plut.cur, ~ Plankt+Pikopl+PO4+NH4, test="F")
Single term additions

Model:
Fotosynt ~ Plankt + Pikopl
       Df Sum of Sq      RSS      AIC F value  Pr(F)
<none>                8.4275 -11.2847
PO4     1    0.2696   8.1579  -9.9350  0.5288 0.4776
NH4     1    0.0259   8.4016  -9.3462  0.0493 0.8272
```

Lisääminen lopetetaan, kun yksikään jäljellä oleva termi ei ole merkitsevä.

Esimerkkimme edustivat kahta perinteistä mallinrakennustapaa: komennolla `add1` *eteenpäin valitsevaa* ("*forward selection*") ja komennolla `drop1` *taaksepäin eliminoivaa* ("*backward elimination*"). Tällä kertaa kumpikin menetelmä johti samaan lopputulokseen, mutta näin ei ole yleensä laita, vaan eri menetelmin päädytään erilaiseen lopulliseen malliin. Yleensä eliminointi (`drop1`) johtaa suurempiin (useampitekijäisiin) malleihin kuin valinta. Tämä johtuu selittävien tekijöiden välisistä riippuvuuksista (epätasapainoisuudesta: §7.11): jokin tekijä on merkitsevä *vain* toisen tekijän kanssa ja tulee valituksi malliin vain jos toinen tekijä on jo valittu malliin. Kumpikaan menetelmä ei välttämättä johda parhaan mahdollisen mallin löytämiseen, koska siihen ei välttämättä ole tietä `drop1` tai `add1` -menetelmillä, vaan meidän pitäisi tutkia useaa mahdollista tekijää yhtaikaa. Havaintoaineistoissa on sangen usein myös samantapaisia tärkeitä tekijöitä, joiden erot voivat olla hyvin pieniä. On enemmänkin sattumaa, kumpi tekijöistä tulee valituksi malliin. Siihen saattaa vaikuttaa jopa valitsemamme tekijänvalintatapa, mutta ainakin otantavaihtelu. Mallinvalinnan perusteella emme siis voi sanoa, että malliin valitsematon tekijä ei ollut tärkeä: malli jossa se olisi ollut mukana saattaa olla melkein yhtä hyvä kuin nykyinen — tai jopa parempi, jos vaihtoehtoisen tekijän valinta olisi johtanut toiseen tienhaaraan ja toiseen lopulliseen malliin.[11]

Eteenpäin valitsevassa menetelmässä voi myös käydä, että jokin aiemmalla askeleella malliin valittu tekijä muuttukiin epämerkitseväksi uusien tekijöiden myötä. *Askeltavassa valinnassa* ("*stepwise regression*") kokeillaan jokaisen `add1`-askelen jälkeen `drop1`-askelta mallissa oleville termeille. Erilaisilla säännöillä pyritään estämään ikuiset silmukat (eli termi näyttää merkitsevältä ollessaan ulkona mallista, mutta muuttuu epämerkitseväksi mallissa). Askeltavat regressiot ovat monien tilastopakettien suosituimpia ohjelmia.

Paketissa `MASS` (Venables & Ripley, 1999) on komento `stepAIC`, joka tekee askeltavan regression AIC:n perusteella. Käyttö on suoraviivaista, mutta tulostus on pitkä, joten näytän vain komennot:

```
> library(MASS)
> plut.1 <- lm(Fotosynt ~ 1)
> stepAIC(plut.1, ~ Plankt+Pikopl+NH4+PO4)
```

Mikä johtaa jälleen samaan lopputulokseen. Komento `stepAIC` toimii sekä lisäävästi että eliminoivasti. Annoimme esimerkissämme lähtökohdaksi mallin, jossa oli vain vakiotermi ja suurimmaksi malliksi kaikki tekijät, joten malli eteni valikoiden eteenpäin. Jos olisimme antaneet komennon

```
>  stepAIC(plut.full)
```

eli suurimman mallimme (ja jättäneet *scope*n pois), `stepAIC` olisi ryhtynyt yksinkertaistamaan mallia.

Olemme kuitenkin tarkastelleet vain yksinkertaista mallia ilman yhdysvaikutuksia. Mikäli määrittelemme suurimman mallin siten, että myös ensimmäisen asteen interaktiot ovat mukana, tuloksemme on toisenlainen:

```
> plut.iso <- lm(Fotosynt ~ (Plankt+Pikopl+NH4+PO4)^2)
> stepAIC(plut.iso)
Start:  AIC= -3.46
 Fotosynt ~ Plankt + Pikopl + NH4 + PO4 + Plankt:Pikopl + Plankt:NH4 +
    Plankt:PO4 + Pikopl:NH4 + Pikopl:PO4 + NH4:PO4

                Df Sum of Sq     RSS     AIC
- NH4:PO4        1    0.0668  5.6661 -5.2245
- Plankt:NH4     1    0.0881  5.6874 -5.1496
- Plankt:Pikopl  1    0.0892  5.6886 -5.1455
- Plankt:PO4     1    0.1476  5.7469 -4.9413
```

[11] *Two roads diverged in a wood, and I — I took the less traveled by; And that has made all the difference.* Robert Frost

```
- Pikopl:NH4     1    0.4697  6.0690 -3.8507
<none>                        5.5993 -3.4617
- Pikopl:PO4     1    1.2824  6.8817 -1.3372

#### ... Tulostusta poistettu ...
#### Tässä lopullinen malli:

             Df Sum of Sq     RSS     AIC
<none>                     6.3597 -8.9150
- Pikopl:NH4  1    0.6805  7.0402 -8.8818
- Pikopl:PO4  1    1.7242  8.0839 -6.1172
- Plankt      1   13.4084 19.7681 11.7667

Call:
lm(formula = Fotosynt ~ Plankt + Pikopl + NH4 + PO4 + Pikopl:NH4 + Pikopl:PO4)

Coefficients:
(Intercept)       Plankt       Pikopl          NH4          PO4   Pikopl.NH4
   -0.20783      0.59332      0.75285     -0.03788      0.31123      0.21538
 Pikopl.PO4
   -0.82396
```

Lopullinen malli on selvästi isompi. Tämä johtuu paljolti siitä, että kriteerinä on AIC eikä F — AIC:n käyttöhän merkitsee samaa kuin "epämerkitsevän" rajan $F = 2$ käyttö. Mukana ovat myös muuttujan `Pikopl` ja ravinnemuuttujien interaktiot. Huomattakoon, että sekä `stepAIC` että `drop1` ymmärtävät sen verran mallinrakennuksen päälle, että ne yrittävät poistaa vain tekijöitä, joille poistaminen on sallittu: sellaista tekijää, joka on jonkin muun tekijän osatekijä ei saa ajatellakaan poistettavaksi. Ensimmäisessä askelessa harkittiin vain interaktiotermien poistamista: päävaikutusta ei voi harkita poistettavaksi ennen kuin on poistettu kaikki interaktiot, joihin se osallinen. Viimeisessä vaiheessa harkittiin poistettavaksi vain termejä `Pikopl:NH4`, `Pikopl:PO4` ja `Plankt`. Muut jätettiin rauhaan, sillä ne sisältyivät mallissa vielä oleviin interaktioihin.

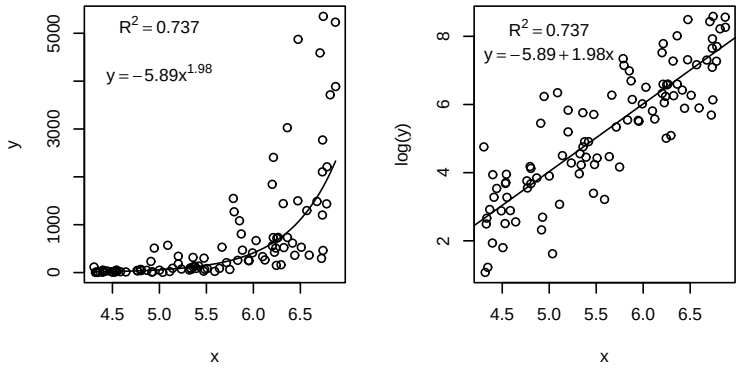
Sama sääntö pätee myös polynomisiin ja muihin vastaaviin regressioihin. Jos meillä on regressiomalli

$$E(y) = b_0 + b_1 x + b_2 z + b_3 x^2 + b_4 z^2 + b_5 xz$$

Niin ensimmäisellä askeleella saamme harkita vain termien x^2, z^2 ja xz poistamista. Saamme harkita termin x poistamista vasta kun sekä x^2 että xz on poistettu. Loogisesti edeten, saamme harkita vakiotermin poistamista vasta kun kaikki muut termit on poistettu. Tästä yleissäännöstä saa poiketa vain hyvin perustellusta syystä silloin kun syntyneellä mallillamme on selvä tulkinta. Esimerkiksi malli $E(y) = b_1 x$ määrittelee regression origon kautta ja voi olla mielekäs kun origolla on absoluuttinen ja mielekäs tulkinta. Tällainen malli voi olla mielekäs, vaikka useimmat näkemäni origon kautta pakotetut regressiot ovat mielettömiä ja virheelliseen päättelyyn pohjautuvia. Malli $E(y) = b_0 + b_2 x^2$ taas määrittelee paraabelin jonka kulmakerroin on 0 pisteessä $x = 0$. Tällainenkin malli voi olla mielekäs, vaikkakin harvoin. Tällainen mallin rakentaminen on hankalampaa, sillä R ei automaattisesti osaa pitää huolta polynomisista riippuvuuksista (ainakaan tätä kirjoitettaessa).

Olen viipynyt mallin rakentamisessa sangen pitkään, sillä aihe on hankala. Vielä tärkeämpi syy on, että sovitettaessa epätasapainoisia, laajoja `lm`-malleja, normaali käytäntö on regressioanalyysin perinteestä kumpuava pikemmin kuin ANOVA-taulukoista. Samoja periaatteita käytetään sekä jatkuville että luokkamuuttujille selittäjinä. *Daphnia*-esimerkissä (§7.10) sovelsimme jo itse asiassa näitä periaatteita: interaktiotermin poistaminen perustui mallien vertailuun ja päätekijän (lämpötila) tunnusluvut arvioitiin tästä pelkistetystä mallista.

Mallin rakentaminen on paitsi vaikeaa myös vastuullista. Tutkija ei voi piiloutua automaattisten proseduurien taakse vaan hänen on otettava vastuu mallistaan. Tämä merkitsee tietystikin tässä luvussa esitettyjen muodollisten sääntöjen noudattamista. Ennen kaikkea se merkitsee kuitenkin,



Kuva 11: Vasemman puoleisessa kuvassa on takaisinmuunnettu lineaarinen regressio, jolle erheellisesti esitetään log–muunnetun lineaarisen regression (oikea puoli) tunnusluvut

että tutkija pyrkii rakentamaan *mielekkään* mallin. Mitä tahansa tekijöitä ei sovi heittää selittäjäksi vain satunnaisen korrelaation vuoksi, vaan lopullisen yhtälön on oltava myös ekologisesti järkevä. Tämä edellyttää yleensä pitkällistä työskentelyä ja harkintaa. On myös muistettava, että tilastollisilla perusteilla meillä voi olla monta suunnilleen yhtä hyvää vaihtoehtoista mallia.

7.13 Kun malli ei sovi: aineiston muunnokset

Ekologeissa tapaa usein merkillistä antautumismielialaa: menetelmän oletuksia rikotaan kuitenkin, joten on sama kuinka analyysit tehdään. Tähän ei pidä alistua, sillä oletusten pätevyyttä pystyy kyllä tarkastelemaan ja mallia parantamaan niin, että tilastollinen päättely varmistuu.

Lineaarisia malleja käytettäessä joudutaan usein tekemään aineiston muunnoksia, joko residuaalien jakauman tai ennusteen muodon korjaamiseksi. Muunnokset ovat sikäli ilkeitä, että ne vaikuttavat sekä virhejakaumaan että regression muotoon, vaikka vain toista haluttaisiin. Esimerkiksi log–muunnos tekee käyrästä eksponentiaalisen ja virheestä log–normaalisen, haluttiin kumpaakin tahi ei. Usein muuttujat muunnetaan takaisin alkuperäiselle asteikolle tuloksia esitettäessä. Tämä voi olla harhaanjohtavaa, sillä mallit on sovitettu muunnetuille arvoille eikä alkuperäisille: ne eivät ole alkuperäisten muuttujien pienimmän neliösumman estimaatteja. Sangen usein näkee kuvia, missä raportoidut korkeat "selitysasteet" (R^2) eivät mitenkään tunnu vastaavan huomattavan suurta hajontaa käyrän suhteen: "selitysaste" on näet arvioitu log–muunnetulle aineistolle, mutta takaisin muuntamisen jälkeen vaihtelu on huomattavasti suurempaa (kuva 11). Takaisinmuunnetut regressiokertoimet ovat myös jossain määrin harhaisia eivätkä vastaisi alkuperäisellä asteikolla arvioituja epälineaarisen mallin kertoimia.

Tutkija pystyy melko usein jo etukäteen arvioimaan, mikä on kohdemuuttujalle sopiva virhejakauma ja mikä voisi olla sopiva varianssin tasoittava transformaatio "ensimmäisenä aproksimaationa":

- Havainnot ovat lukumääriä eli nollia tai positiivisia kokonaislukuja eikä niillä ole ehdotonta maksimia. Odotettu jakauma on tällöin *Poisson* ja varianssi on suhteessa odotusarvoon. Tällöin vakauttava muunnos on *neliöjuuri*.
- Havainnot ovat osuuksia kokonaismäärästä. Ne ovat nollia tai positiivisia kokonaislukuja, mutta niillä on ehdoton katto: kaikki paikat ovat täynnä, kaikki koekasvit tutkittu. Myös prosenttilukuja voidaan tarkastella tällaisina lukuina, sillä ne on saatu vastaavasti kokonaisluvuista. Odotettu jakauma on tällöin *binominen* ja varianssi suurimmillaan 50 % kohdalla. Vakauttava muunnos on *arcsin neliöjuuri*.

- Havainnot ovat pitoisuuksia eli nollaa suurempia reaalilukuja (nolla saattaa joskus esiintyä aineistossa, mutta se johtuu siitä, että mittalaite ei ollut kyllin herkkä). Odotettu jakauma on tällöin *gamma* ja keskihajonta on suhteessa odotusarvoon. Vakauttava muunnos on *logaritminen*.

Mallin sovituksen jälkeen on syytä tarkastella jakauman ja mallin sopivuutta. Sangen usein tilastollinen analyysi aloitetaan tarkastelemalla muuttujien jakaumaa ja tekemällä tarpeelliseksi katsottavia muunnoksia. Tämä toimii usein oikein hyvin, mutta mallit olettavat mallivirheiden (residuaalien) jakautuvan oletustensa mukaan, joten jakaumaoletuksia voi todella tarkastella vain suhteessa malliin. Jakauman normaalisuuden tutkiminen on kenties saanut liian suuren painon ekologien parissa. Hyvin usein tutkimme kuitenkin keskiarvoja (joita parametrikertoimetkin ovat), joiden jakauma on aina normaalinen suurilla aineistoilla riippumatta havaintojen jakaumasta. Pienillä aineistoilla taas emme voi sanoa jakaumista juuri mitään. Jakaumien sijaan on syytä tarkastella muita patologisia piirteitä:

- Residuaalien "tasainen" eli ei-systemaattinen jakautuminen: Lineaariset mallit olettavat mm., että varianssi on riippumaton keskiarvosta, eli käytännössä yhtä suuri kaikissa luokissa ja kaikissa osissa regressiota (eli virhevaihtelu on *homoskedastista*). Jos tämä oletus ei pidä paikkaansa, voi regressiokertoimien estimointi olla tehotonta ja mallin antamien ennusteiden luottamusvälit joko liian kapeita tai liian leveitä.
- Selittävän mallimme ekologinen mielekkyys ja järkevältä näyttävä regressiosuoran muoto.
- Yksittäisten havaintojen vaikutus: mikäli regressiomme määräytyy enimmäkseen yhden vaikuttavan pisteen perusteella, se ei vaikuta kovin luotettavalta. Jos "merkitsevä" testituloksemme johtuu vain yhden pisteen paremmasta selittämisestä, se ei ole kovin uskottava.

Pyrkimys jakaumien normalisointiin saattaa vahingossa korjata nämäkin ongelmat, mutta usein parempaan tulokseen päästään syöksymällä suoraan ongelman eikä sen oireiden kimppuun.

Lineaaristen mallien oletuskuva (`plot.lm`, esim. kuva 6) on mainio keino tutkia mallin sopivuutta. Huomattakoon, että useimmat jakauman sopivuutta tutkivat testit ovat "asymptoottisesti merkitseviä" eli aineiston koon kasvaessa ne pystyvät löytämään yhä pienempiä eroja havaitun ja odotetun jakauman välillä. Regressiodiagnostisen kuvien epämuodollinen tarkastelu saattaa olla valaisevampaa kuin muodollinen testaaminen.

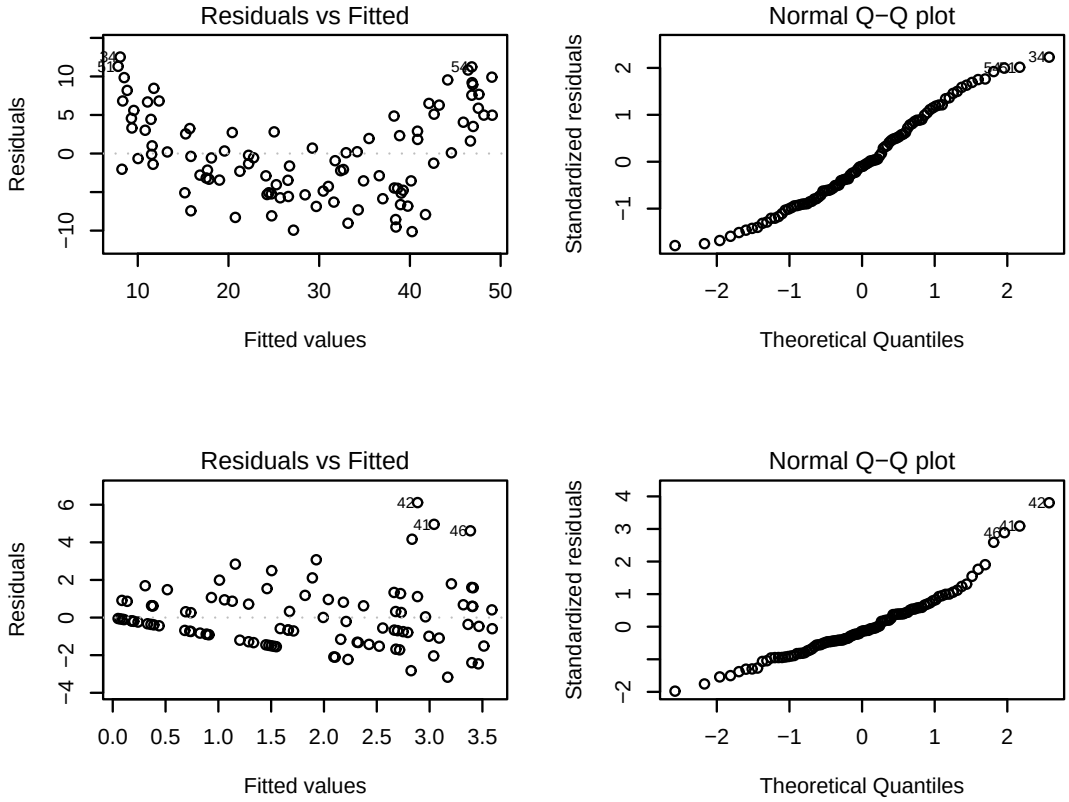
Malli voi olla huono kahdella päätavalla (kuva 12):

- Regression muoto on väärä.
- Residuaalien jakauma on väärä.

Näitä kahta tapausta voi olla vaikea erottaa. Usein hyödyllisintä on tarkastella *Residuals vs. Fitted* osakuvaa `plot.lm`-kuvassa. Jos sekä muoto että jakauma ovat oikeat, residuaalit muodostavat tasaisen vyön oikean arvon ($y = 0$) ympärille. Jos residuaaleissa näkyy jokin selkeä kuvio kuten kaari, regression muoto on todennäköisesti väärä. Jos taas residuaalit ovat keskimäärin melko oikeita, mutta ne laajenevat viuhkamaisesti (tai salmiakkimaisesti keskeltä), virhejakauma on todennäköisesti väärä. Kumpikin voi näkyä Q–Q–plotissa epälineaarisena, mutta luonnollisestikin väärän virhejakauman pitäisi näkyä selkeämmin. Vaikka ensimmäiseksi mieleen tuleekin muuntaa y–muuttujaa virheen korjaamiseksi, on toki muistettava, että malli voi korjaantua myös regressiomallia muuttamalla (kuten meille kävi luvun 7.3 esimerkissä) muuntamalla selittäviä tekijöitä.

On myös mahdollista kysyä aineistolta itseltään, mikä on mallin sopivuuden parhaiten takaava muunnos käyttämällä Box–Cox -muunnosta y^λ. Parametria λ varioimalla pystymme tekemään monta suosittua muunnosta:

$\lambda = -1$	käänteismuunnos
$\lambda = 0$	logaritmimuunnos
$\lambda = \frac{1}{2}$	neliöjuurimuunnos
$\lambda = 1$	ei muunnosta
$\lambda = 2$	neliömuunnos



Kuva 12: Ylemmässä kuvaparissa regression muoto on väärä (mutta virhejakauma oikea), alemmassa kuvassa virhejakauma on väärä (mutta muoto on oikea).

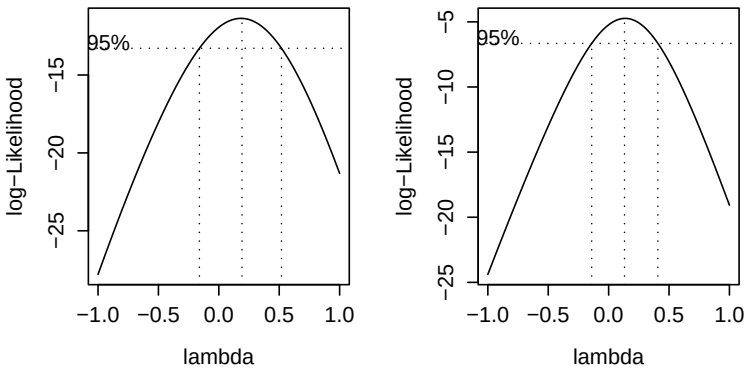
Box–Cox -muunnoksessa tarkastellaan mallin hyvyyden riippuvuutta parametrista λ, ja näin saada selville mikä olisi paras muunnos kyseisesssä mallissa. Lisäksi saadaan selville, mitkä muunnokset ovat likipitäen ("ei-merkitsevästi") yhtä hyviä. Jos $\lambda = 1$ sisältyy muunnoksen luottamusväliin, meillä ei ole syytä muuntaa aineistoa. Vaikka tuloksena onkin desimaaliluku λ, muunnokseksi valitaan yleensä jokin sopiva tasainen luku, esim. yllä olevasta asetelmasta. Box–Cox -muunnos ei välttämättä pyri korjaamaan virhejakaumaa, vaan se pyrkii korjaamaan myös regression muodon siten että uskottavuus maksimoituu.

Paketissa `MASS` (Venables & Ripley, 1999) on komento `boxcox` joka piirtää λ–profiilin ja merkitsee siihen 95 % luottamusvälit (kuva 13):

```
> library(MASS)
> par(mfrow=c(1,2))
> boxcox(formula(plut.lm),data=plutakko, lambda=seq(-1,1,len=20))
> boxcox(formula(plut.new),data=plutakko, lambda=seq(-1,1,len=20))
```

Jos meillä on vielä jäljellä mallitulokset olioissa `plut.lm` ja `plut.new` (§7.3), voimme eristää niiden mallilausekkeet komennolla `formula` — muutoin mallilausekkeet on kirjoitettava uudestaan.

Kumpikin malli näyttäisi vaativan muunnosta (kuva 13), vaikka alkuperäisessä analyysissämme (§7.3) olimmekin tyytyväisiä lopullisen mallin diagnostisiin kuviin. Koska $\lambda = 0$ sisältyy luottamusväleihin, log–muunnos voisi olla sopiva. Tämä tietysti sopisi myös apriorisesti, sillä fotosynteesin intensiteetti on eräänlainen pitoisuus, jonka voi kuvitella olevan gamma–jakautunut. Toinen mahdollinen muunnos voisi olla kuutiojuuri ($\lambda = \frac{1}{3}$), jolle myös on fysikaalinen tulkinta: klorofyllipatjan paksuus.



Kuva 13: Box–Cox -profiilit kalliolammikoiden fotosynteesiaineiston alkuperäiselle (Ranta *et al.*, 1989) ja lopulliselle analyysille (§7.3).

Tässä kappaleessa lähtökohtanamme on ollut, että jos tilastollinen malli ja havainnot ovat ristiriidassa, muutetaan havaintoja malliin sopiviksi. Vaihtoehtona on muuttaa mallia havaintoihin sopivaksi. Tässä johdattelevessa oppaassa emme ehdi käsitellä paremmin aineistoihin sopivia tilastollisia malleja, mutta tässä eräitä vaihtoehtoja (R-intro, 2000; Venables & Ripley, 1999):

- Yleistetyt lineaariset mallit (komento `glm`) perivät kaikki lineaaristen mallien ominaisuudet, mutta ovat yleisempiä. Virhejakauma voi olla myös epänormaalinen (Poisson, binominen, gamma). Lisäksi ne käyttävät *linkkifunktiota*, jolla *lineaarinen ennuste* voidaan muuntaa näennäisen epälineaariseksi. Näin niiden avulla voi väistää yhden aineiston muunnoksen ongelman: regression muodon ja virhejakauman voi valita itsenäisesti. Yleistetyt lineaariset mallit ovat tulleet erittäin suosituksi ekologiassa. Perusviite on McCullagh & Nelder (1989). Ekologeille kirjoitettu johdanto on Crawley (1993). Yleistettyjen lineaaristen mallien eräitä laajennuksia varten, jotka soveltuvat keskenään korrelovia havaintoja sisältäviin aineistoihin tai niin kiinteitä kuin satunnaisiakin vaikutuksia sisältävien mallien analysointiin, on R:ssä käytettävissä mm. paketit `gee` (*generalized estimating equations*), `GLMMGibbs` (*generalized linear mixed models by Gibbs sampling*) ja `MASS`-paketin funktio `glmmPQL` (*GLMM via Penalized Quasi-Likelihood*).

- Epälineaarinen regressio sopii, jos meillä *apriorisen teorian mukaan* on jokin epälineaarinen funktio, jonka haluamme sovittaa aineistoon. Menetelmää ei voi käyttää muodon etsimiseen, vaan ainoastaan, kun *tiedämme* tarvitsevamme tiettyä epälineaarista mallia (kuten Michaelis–Menten, Gompertz). R:ssä ne sovitetaan yleensä pienimmän neliösumman malleina (paketit `nls`, `nlme`), mutta myös suurimman uskottavuuden menetelmät muilla virhejakaumilla ovat mahdollisia (komento `nlm`). Mallien sovittamien voi olla hankalaa. Venables & Ripley (1999) käsittelevät pitkään näitäkin malleja.

- Tasoittavat funktiot ja muut "*modernin regression*" ihmeet. Näissä ei regression muotoa määritetä funktiolla, vaan menetelmät pyrkivät itse etsimään tasaisen muodon. Käyttäjä voi säädellä muodon tasaisuutta tai herkkyyttä paikallisille piirteille, mikä vastaa parametrien määrän valintaa lineaarisessa regressiossa. Kuuluisin näistä menetelmistä on S-PLUS–ohjelman yleistetty additiivinen malli (`gam`), jonka parannettu versio on R-paketissa `mgcv`. Monia muita "modernin regression" ihmeitä on tarjolla paketeissa `modreg` (moderni regressio), `polymars` (adpatiiviset regressiosplinit), `nnet` (neuroverkot) ja `trees` (luokittelu- ja regressiopuut). Perusviite on Hastie & Tibshirani (1990), mutta myös Venables & Ripley (1999) käsittelevät monia menetelmiä.

Viitteet

Crawley, M. J. (1993): *GLIM for ecologists.* Blackwell, Oxford, 379 ss.

Hastie, T. J. & Tibshirani, R. J. (1990): *Generalized additive models.* Chapman & Hall, London, 335 ss.

McCullagh, P. & Nelder, J. A. (1989): *Generalized linear models.* Chapman & Hall, London, 511 ss.

Pinheiro, J. & Bates, D. (2000): *Mixed-effect models in S and S-PLUS* Springer.

R-intro (2000): An introduction to R. R Development Core Team.

Ranta, E., Rita, H. & Kouki, J. (1989): *Biometria.* Yliopistopaino, Helsinki.

Venables, W. & Ripley, B. (1999): *Modern applied statistics with S-PLUS.* Springer, kolmas painos.