Luku 6
Yleinen mallin sovitus

Tässä kappaleessa käsitellään lyhyesti yleisen mallin sovitusta. Muuttujien välille oletetaan parametrisoitavissa oleva relaatio, ja parametrien arvot määritetään ottaen huomioon kaikissa suureissa esiintyvät epätarkkuudet. Mallinsovitus perustuu ansiofunktioon, jonka arvot määräävät tuntemattomien parametrien arvot ja niiden virherajat. Parametrit ratkaistaan maksimoimalla mallin uskottavuutta (*maximum likelihood*–estimaatit). Yleisessä, epälineaarisessa tapauksessa on laskennassa turvauduttava iteratiivisiin menetelmiin ansiofunktion minimoinnissa. Tähän voidaan käyttää kuitenkin mitä tahansa optimointimenetelmää. Tällä kurssilla ei käsitellä varsinaisesti numeriikkaa (tätä varten on kurssi Tähtitieteen numeeriset menetelmät). Tähän tehdään poikkeus seuraavassa luvussa, jossa käsitellään epälineaarista pienimmän neliösumman ongelmaa.

Mallit voisi jakaa eri ryhmiin lineaarisuuden ja virhemallin mukaan. Luvun 4 lineaariset mallit käsittelivät ainoastaan havaintoja, joiden virheet noudattivat normaalijakaumaa. Tällöin oli jo esillä painotettu sovitus, jossa pisteiden havaintotarkkuudet otettiin myös huomioon.

Tässä luvussa puhutaan yleisistä malleista, jotka eivät enää ole lineaarisia määritettävien parametrien suhteen. Samoilla menetelmillä voidaan toki suorittaa lineaaristen mallien sovitus - jos joka tapauksessa oletetaan normaalijakautuneet virheet, ovat luvun 4 menetelmät huomattavasti nopeampia, ja lisäksi saadaan ohjelmasta suoraan esim. parametrien luotettavuusvälit. Sama pätee epälineaarisiin malleihin: jos virheet ovat normaalijakautuneita, seuraavan luvun menetelmät on tavallisesti nopeampaa kuin yleisen optimointirutiinin käyttö. Lisäksi rutiinit pystyvät (periaatteessa) laskemaan automaattisesti luotettavuus- ja ennustevälit.

6.1 Epälineaarinen pns–ongelma

Seuraavassa tarkastellaan erikseen epälineaarista pienimmän neliösumman ongelmaa, jossa optimoinnissa voidaan käyttää hyväksi minimoitavan funktion muotoa. Malli on epälineaarinen parametrien suhteen, joten lineaaristen mallien tapauksessa esiteltyjä kaavoja ei nyt voida soveltaa. Esimerkkinä voisi olla esimerkiksi mallin $y = x^a$ sovittaminen, jossa a on määritettävä parametri (tämä nimenomainen ongelma voitaisiin palauttaa lineaariseksi ottamalla yhtälöstä puolittain logaritmi - mutta miten silloin kävisi mittausarvojen virhejakaumien?).

Minimoitavana oleva funktio on muotoa pienimmän neliösumman tapauksesa

$$\chi^2(\vec{x}) = \sum_{i=1}^{M} [f_i(\vec{x})]^2. \tag{6.1}$$

Jacobin matriisin, $J(\vec{x}) = \partial f_i/\partial x_j$, avulla lausuttuna minimoitavan funktion gradientti, ∇F, ja Hessin matriisi, H ovat

$$\begin{aligned} \nabla\chi^2(x) &= 2\,J(\vec{x})^T f(\vec{x}) \\ H(\vec{x}) &= 2\,J(\vec{x})^T J(\vec{x}) + 2\sum_{i=1}^{M} f_i(\vec{x})\nabla^2 f_i(\vec{x}). \end{aligned}$$

Lähellä minimiä χ^2-funktio voidaan approksimoida neliöllisellä funktiolla

$$\chi^2(\vec{a}) \approx \vec{c} + \overrightarrow{\nabla\chi^2}\cdot\overrightarrow{da} + \frac{1}{2}\overrightarrow{da}\cdot H\cdot\overrightarrow{da}, \tag{6.2}$$

jonka minimi löytyy suoraan yhdellä askeleella

$$\vec{a}^{\,i+1} = \vec{a}^{\,i} + H^{-1}\big[-\nabla\chi^2(\vec{a}^{\,i})\big]. \tag{6.3}$$

Jos olemme kaukana minimistä, neliöllinen approksimaatio ei ole kovin hyvä. Sen sijaan voimme ottaa askeleen jyrkimmän suunnan menetelmän mukaisesti

$$\vec{a}^{\,i+1} = \vec{a}^{\,i} - \mu\nabla\chi^2(\vec{a}^{\,i}), \tag{6.4}$$

missä μ on jokin vakio.

Algoritmissa tarvitaan χ^2-funktion ensimmäiset derivaatat parametrien suhteen ($\rightarrow$ gradientti) sekä vastavat toiset derivaatat ($\rightarrow$ Hessin matriisi). Nämä voidaan laskea suoraan χ^2 määritelmästä.

$$\frac{\partial\chi^2}{\partial a_k} = -\,2\sum_{i=1}^{N}\frac{(y_i-\overline{y}\,)}{\sigma_i^2}\,\frac{\partial y}{\partial a_k}$$

$$\frac{\partial^2\chi^2}{\partial a_k\,\partial a_l} = 2\sum_{i=1}^{N}\frac{(\partial y/\partial a_k)\,(\partial y/\partial a_l) - (y_i-\overline{y}\,)\,\frac{\partial^2 y}{\partial a_l\partial a_k}}{\sigma_i^2}$$

Määrittelemällä

$$\begin{aligned} \beta_k &= -\,\frac{1}{2}\frac{\partial\chi^2}{\partial a_k} \\ \alpha_{kl} &= \frac{1}{2}\frac{\partial^2\chi^2}{\partial a_k\partial a_l} \end{aligned}$$

saadaan yhtälö (6.3) mukaisesti minimi ratkaistua yhtälöryhmästä

$$\sum_{l=1}^{M}\alpha_{kl}\delta a_l = \beta_k \tag{6.5}$$

ja jyrkimmän suunnan menetelmä (6.4) tulee puolestaan muotoon

$$\delta a_l = \mu\beta_l. \tag{6.6}$$

Edellä δa kuvaa iteraatiokierroksella parametriarvoon tehtävää korjausta.

Levenberg-Marquardt on ehkä yleisin pienimmän neliösumman ongelmien ratkaisussa käytetty algoritmi. Siinä käytetään aluksi jyrkimmän suunnan menetelmää, mutta iteraatioiden edetessä otetaan huomioon myös Hessin matriisi. Lisäksi Hessin matriisia käytetään arvioitaessa askelpituutta μ, jolloin jyrkimmän suunnan menetelmän yhtälö (6.6) korvataan yhtälöllä

$$\delta a_l = \frac{1}{\lambda \alpha_{ll}} \beta_l \iff \lambda \alpha_{ll} \delta a_l = \beta_l, \tag{6.7}$$

missä λ on jokin luku. Menetelmät yhdistetään määrittelemällä uusi matriisi

$$\alpha'_{jj} = \alpha_{jj}(1+\lambda),$$
$$\alpha'_{jk} = \alpha_{jk}, \quad j \neq k.$$

Yhtälössä (6.5) matriisi α korvataan nyt matriisilla α'. Kun λ on suuri, lähestyy yhtälö jyrkimmän suunnan menetelmää (6.7).

Varsinainen algoritmi on seuraava:

1. laske $\chi^2(\vec{a}\,)$ ja aseta parametrille λ pieni arvo, esim. $\lambda = 0.001$
2. ratkaise $\delta\vec{a}$ yhtälöstä
$$\sum_{j=1}^{M} \alpha'_{kj} \delta a_j = \beta_k \tag{6.8}$$
ja laske $\chi^2(\vec{a} + \delta a)$
3. jos χ^2-arvo kasvoi, kasvata λ-arvoa - muussa tapauksessa pienennä λ:aa ja päivitä $\vec{a}$
4. palaa kohtaan 2

Yleensäkin optimointirutiinit tarvitsevat mieluusti ainakin optimoitavan funktion derivaatat. Kun käytetään pns-rutiinia, nämä on jo koodattu valmiiksi ohjelmaan. Epälineaarisen ongelman ratkaisu tapahtuu aina iteratiivisesti, joten ratkaisun laskeminen voi kestää *huomattavasti* pidempään kuin lineaaristen mallien tapauksessa.

6.2 Epälineaarinen pns-ongelma *R*-ohjelmistossa

R-ohjelmistossa epälineaaristen pienimmän neliösumman ongelmien ratkaisemiseen tarkoitettuja rutiineja löytyy ainakin kirjastosta `nls` (*non-linear least squares*), ja täreimmän rutiinin nimi on myös **nls**. Argumentteina ovat mm. käytetty data-aineisto sekä sovitettava malli. Optionaalisen parametrin `control` avulla voidaan asettaa mm. suurin sallittu iteraatioiden määrä ja vaadittu toleranssi. Rutiini palauttaa `nlsModel` tyyppisen objektin. Tämä sisältää mm. sovitetun mallin yhtälön, sovitetut parametriarvot, residuaalivektorin ym. Rutiinilla `predict` voidaan laskea mallin ennusteet halutuissa pisteissä - jos parametriksi antaa ainoastaan sovitetun mallin, ennusteet lasketaan havaintopisteille.

Periaatteessa rutiinin `nls` pitäisi pystyä laskemaan myös mallin luotettavuusvälit ja ennustevälit. Valitettavasti näitä ominaisuuksia ei kuitenkaan ole vielä toteutettu. Sovitettujen parametrien arvioidut virherajat saa näkyviin komennolla `summary`.

Esimerkki 6.1. Tarkastellaan *R*-ohjelmiston mukana tullutta auringonpilkkudataa. Alkuperäinen tiedosto on aikasarja, josta otetaan tarkasteltavaan vektoriin ainoastaan osa. Vektori sisältää yhden havaintopisteen kuukautta kohden. Havaintoihin sovitetaan epälineaarinen malli

$$y(t) = (1 + \sin(at + b))(c + dt),$$

missä t on kuukauden indeksi ja $a - d$ mallin parametreja. Seuraavat komennot suorittavat pns sovituksen ja piirtävät sovitetun mallin sekä sovituksen jäännöspoikkeamat.

Huom. Nykyään sijoitusopetaattori on R-ohjelmistossa kirjoitetta '<-', ei '_'.

```
data(sunspots) ;
spots _ sunspots[1520:2600] ; # vektori: yksi piste/kk
library(nls)
new _ data.frame(T=1:length(spots), Y=spots) ;

p1 _ nls(Y ~ (1.0+sin(T*a+b))*(c+d*T),
       data = new,
      start = list(a=0.048,b=1.6,c=30.0,d=0.01),
      trace = TRUE,
     control= list(maxiter=200, tol=1.0e-5));

# piirretään sovitus
plot(spots, main=p1$m$formula()) ;
lines(predict(p1), col="blue", lwd=2)

# piirretään residuaalit
lines(p2$m$resid(), col="orange")

# jaksonpituudet vuosina:
cat("eka = ", 2*pi/(12*p1$m$getAllPars()[1]), " vuotta˙") ;
```

Edellinen malli on ainoastaan esimerkki pns sovituksesta – se ei ole vielä kovinkaan hyvä malli käytetyille havainnoille (esim. todellinen jaksonpituus ei ole vakio ja sykli ei tarkalleen ottaen ole sinimuotoinen). Kokeile esim. mallia, jossa jaksonpituus muuttuu lineaarisesti havaintosarjan aikana.

6.3 Yleinen malli

Seuraavaksi käsitellään täysin yleistä mallia. Ensinnäkin malli voi tietenkin olla ei-lineaarinen, ja toisaalta havaintojen virhejakaumat voivat olla mielivaltaisia (mutta tunnettuja).

Jatketaan vielä hetken jakoa riippumattomiin ja riippuviin muuttujiin, ja merkitään mallia aluksi $f(\vec{x};\vec{a})$, missä vektori $\vec{a}$ sisältää mallin parametrit ja x on pisteen koordinaatti. Havainnot koostuvat vastaavasti pisteistä y_i, joilla oletetaan olevan tunnettu todennäköisyysjakauma. Merkitään i:nnen pisteen todennäköisyystiheysfunktiota p_i. Mallin todennäköisyys, kun pisteessä x_i on havaittu arvo y_i on $p_i(f(x_i);\vec{a})$ (*itse asiassa tämä olisi havaintojen todennäköisyys, kun malli on kiinnitetty mutta Bayesin kaavan perusteella nämä ovat sama asia, kun käytettävissä ei ole muuta informaatiota*). Tässä on kirjoitettu $\vec{a}$ mukaan osoittamaan todennäköisyysjakauman riippuvuutta mallin parametreista. Kaikki pisteet mukaan laskettuna havaintojen todennäköisyys on

$$p \propto \Pi_{i=1}^{N} p_i(f(x_i)|\vec{a}), \tag{6.9}$$

ja parametrit $\vec{a}$ saadaan maksimoimalla edellistä lauseketta. Näin johdettua $\vec{a}$:n arvo on *maximum likelihood* eli *ML*-estimaatti, suomeksi **suurimman luotettavuuden estimaatti**. Edellisestä lausekkeesta voidaan ottaa logaritmi sen maksimikohdan muuttumatta, joten yhtä hyvin maksimoida summalauseketta

$$\ln p = \sum_{i=1}^{N} \ln p_i(f(x_i),\vec{a}). \tag{6.10}$$

Todennäköisyystiheysjakaumat p_i on luonnollisesti tunnettava. Jakaumia voidaan approksimoida jollakin tunnetulla jakaumamuodolla (normaalijakauma, Poisson, ...) esim. teoreettisen päättelyn pohjalta. Joskus empiirinen jakauman muoto voidaan johtaa suoraan havainnoista.

Esimerkki 6.2. Etsitään *ML*-estimaatti lukujen keskiarvolle, kun kunkin pisteen virhejakauma on $N(0,\sigma_i)$. *ML*-estimaatin laskenta ei tietenkään edellytä gaussisia muuttujia, ja estimaatti voitaisiin laskea samaan tapaan minkä tahansa virhejakauman tapauksessa. Kirjoitetaan todennäköisyyden lauseke

$$p \propto \Pi_{i=1}^{N} p_i(\mu) = \Pi_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma_i} \exp[-\frac{1}{2}(\frac{x_i-\mu}{\sigma_i})^2].$$

Otetaan lausekkeen logaritmi, jolloin maksimoitavaksi lausekkeeksi tulee

$$\begin{aligned}
&\max\{\ln(\Pi_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma_i} \exp[-\frac{1}{2}(\frac{x_i-\mu}{\sigma_i})^2])\} \\
&\sim \max\{\sum \ln[\frac{1}{\sigma_i}\exp[-\frac{1}{2}(\frac{x_i-\mu}{\sigma_i})^2]\ ]\} \\
&\sim \max\{\sum \ln\frac{1}{\sigma_i} + \sum (-\frac{1}{2})(\frac{x_i-\mu}{\sigma_i})^2\} \\
&\sim \min\{\sum (\frac{x_i-\mu}{\sigma_i})^2\},
\end{aligned}$$

mikä on tietenkin tavallinen pienimmän neliösumman lauseke. Pienimmän neliösumman ratkaisu on siis ML-estimaatti – mutta vain jos virhejakauma on normaali!

Tässä tapauksessa lausekkeen minomoiva μ:n arvo voidaan laskea analyyttisesti. Minimikohdassa lausekkeen derivaatta on nolla,

$$D(\sum (\frac{x_i-\mu}{\sigma_i})^2) = \sum (-\frac{2}{\sigma_i})\frac{x_i-\mu}{\sigma_i} = -2\{\sum \frac{x_i}{\sigma_i^2} - \mu \sum \frac{1}{\sigma_i^2}\} = 0,$$

joten keskiarvon *ML*-estimaatiksi saadaan

$$\mu_{\mathrm{ML}} = \frac{\sum x_i/\sigma_i^2}{\sum 1/\sigma_i^2},$$

eli kyseessä on varianssin käänteisarvolla painotettu keskiarvo.

Osoita, että tämän estimaatin jakauma on myös normaali, ja että sen keskihajonta on

$$\sigma(\mu_{\mathrm{ML}}) = (\sum \frac{1}{\sigma_i^2})^{-1/2}.$$

Aiempien lukujen lineaarisissa malleissa tehtiin jako riippumattomiin (x) ja riippuviin muuttujiin (y), ja huomioon otettiin ainoastaan y:n havaintovirheet. Yleisemmin kukin havainto muodostaa yhden pisteen mahdollisesti useampiulotteisessa avaruudessa ja kaikki muuttujat ovat samanarvoisessa asemassa. Maksimoitava lauseke on parempi kirjoittaa muotoon

$$p \propto \Pi_{i=1}^{N} p_i(\vec{x}_i|\vec{a}), \tag{6.11}$$

jossa summattava termi tarkoittaa havaintopisteen $\vec{x}$ todennäköisyyttä, kun malli on tiedossa ja sen parametrivektori $\vec{a}$ on kiinnitetty.

Aina ei estimaatin lauseke ratkea helposti analyyttisesti kuten esimerkissä 6.2. Numeerinen ratkaisu vaatii vähintään maksimoitavan lausekkeen kirjoittamista, ja mahdollisesti hyvän alkuarvon antamista optimointirutiinille. Useimmat optimointirutiinit käyttävät myös funktion derivaattoja. Jos näitä ei osata laskea analyyttisesti, voi derivaatat laskea numeerisesti differenssien avulla. Numeeriset differenssit hidastavat merkittävästi laskentaa ja tekevät siitä epästabiilimman – niitä ei siis ole syytä käyttää, ellei ole pakko.

Esimerkki 6.3. *R*-ohjelmisto sisältää yleistä optimointia varten rutiinin `optim`. Parametreina ovat mm. minimoitava funktio, funktion derivaatta, parametrien alkuarvot sekä kovergenssiehtoihin liittyviä parametreja. Myös käytetty optimointimenetelmä voidaan valita ja käytettävissä ovat kvasi-Newton menetelmä `BFGS`, konjugaattigradienttimenetelmä `CG`, simuloitu jäähdytysmenetelmä `SANN` sekä Nelder-Mead simplex algoritmi, joka on myös oletusarvo. Ainoastaan kaksi viimeistä eivät tarvitse funktion derivaattoja. Muidenkin menetelmien tapauksessa derivaattafunktioita ei ole pakko antaa, mutta silloin derivaatat lasketaan numeerisesti.

Tehdään sama mallinsovitus kuin esimerkissä 6.2 - tällä kertaa yleisellä optimointirutiinilla.

```
# sovitettava funktio
fun _ function(t, para) {
   (1.0+sin(para[1]*t+para[2]))*(para[3]+para[4]*t) ;
}
# ansiofunktio - minimoidaan chi2
chi2 _ function(para) {
   summa = 0.0 ;
  for(t in 1:length(spots)) {
     summa = summa + (fun(t,para)-spots[t])^2.0 ;
  }
  summa ;
}
#
a _ optim(c(0.048, 2.50, 18.0, 0.065),    # alkuarvot
          chi2, method="SANN",            # menetelmä
          control=list(maxit=20000,
                       temp=0.5,
                       parscale=c(0.005, 0.2, 2.0, 0.01),
          fnscale=+1.0e6) ) ;
```

Ohjelma löytää saman ratkaisun kuin esimerkin 6.1 pns-rutiini – tosin pari kertaluokkaa hitaammin. Edellä myös minimoitiin suoraan χ^2 funktiota. Jos olisi yritetty maksimoida suoraan todennäköisyyksien tuloa, laskenta olisi kaatunut numeerisiin ongelmiin. Jos todennäköisyysjakaumaksi oletetaan esim. Cauchyn jakauma, pitää edellinen ansiofunktio `chi2` korvata *maksimoitavalla* rutiinilla.

```
ansio _ function(p)  {
    tulo = 1.0 ;
    for(t in 1:length(spots)) {
       tulo = tulo * pcauchy(fun(t,para), spots[t], scale[t]) ;
    }
    tulo ;
}
```

Minimointi muutetaan maksimoinniksi vaihtamalla parametrin `fnscale` etumerkkiä (tai yhtä hyvin funktion etumerkkiä). Tässä on oletettu, että kunkin havainnon virhe on annettu jakauman skaalaparametrien `scale[]` kautta.

Edellä käsiteltiin aikaa riippumattomana muuttujana, jonka mahdollisia virheitä ei otettu huomioon. Yleensä kaikkien muuttujien virheet on otettava huomioon ansiofunktion lausekkeessa. Myöhemmin seuraavat esimerkit käsittelevät tällaisia tapauksia.

Optimointirutiinilla voidaan määrittää suoraan parametrien arvot, mutta parametrien virheiden estimointi on vaikeampaa. Virhearviot voi määrittää esimerkiksi seuraavassa luvussa kuvatuilla menetelmillä, parhaiten ehkä suoraan Monte Carlo menetelmällä.

6.4 Virhearvioiden määrittämisestä

Useimmat pienimmän neliösumman sovituksessa käytetyt algoritmit antavat myös arviot sovitettujen parametrien virheiksi. Virhearviot perustuvat χ^2-funktion paikalliseen käyttäytymiseen, eli käytännössä **funktion derivaattoihin** parametrien suhteen löydetyssä pisteessä. Virhearvio voi olla virheellinen, jos kyseinen funktio ei käyttäydy säännöllisesti minimikohdan läheisyydessä. Saatu virhearvio on yksi ainoa luku, joten sen ei sisällä tietoa virhejakauman muodosta. Jos virheiden oletetaan olevan normaalijakautuneita tämä ehkä riittääkin - määrittelevähän keskiarvo ja keskihajonta jakauman täydellisesti.

Jos havaintopisteille ei ole käytettävissä virhearvoita, ei niitä myöskään voida suoraan käyttää sovitettujen parametrien virhearvioita arvioitaessa. Tällöin voidaan turvautua esim. **Bootstrap**-menetelmään, jossa saatua havaintoaineistoa käytetään määrittämään muuttujan todellinen jakauma. Menetelmä ei vaadi mitään etukäteisoletuksia havaintojen jakaumasta – jos jakauma tunnetaan entuudestaan, ovat muut menetelmät kuitenkin tarkempia parametrien virheitä arvioitaessa. Olkoon havaintopisteiden lukumäärä N. Havainnoista muodostetaan uusi, yhtäsuuri N pisteen otos siten, että siihen lisätään N kertaa sattumanvaraisesti alkuperäisistä havainnoista valittu piste. Sama havainto voi siis esiintyä otoksessa useammankin kerran. Mallin parametrit määrätään otoksen perusteella ja operaatio toistetaan, jolloin tuloksista voidaa päätellä mallin parametrien virheet. Tasapainotetussa bootstrap-menetelmässä (*balanced bootstrap*) jokainen havaintopiste esiintyy otoksissa täsmälleen yhtä montaa kertaa (siis kaikki otoksen yhteen laskettuna). Olkoon $\hat{\theta_i}$ estimaatti, joka on määrätty i:nnestä otoksesta. Varsinainen bootstrap estimaatti on

$$\hat{\theta}_B = \frac{1}{P}\sum_{i=1}^{P} \hat{\theta}_i .$$

Bias määritellään parametrin odotusarvon virheenä $b = E[\hat{\theta}] - \theta$. Odotusarvo saadaan edellisestä kaavasta ja parametriarvo θ korvataan *koko aineistosta* määritetyllä parametriarvolla $\hat{\theta}_0$, eli biaksen estimaatti on $\hat{b} = \hat{\theta}_B - \hat{\theta}_0$. Tämän avulla voidaan kirjoittaa bias-korjattu estimaatti

$$\hat{\theta}_0 - \hat{b} = 2\hat{\theta}_0 - \hat{\theta}_B .$$

Olettaen, että parametrin jakauma noudattaa normaalijakaumaa, luotettavuusvälit voidaan arvioida suoraan variassin avulla,

$$\sigma^2(\hat{\theta}) = \frac{1}{P-1}\sum_{i=1}^{P} (\hat{\theta_i} - \hat{\theta}_B)^2 .$$

Ottaen huomioon bias-korjauksen, saadaan 100(1-α) % luotettavuusväliksi

$$(2\hat{\theta}_0 - \hat{\theta}_B) \pm z_{\alpha/2}\,\sigma(\hat{\theta})\,,$$

missä z-arvot saadaan normaalijakaumasta: $\mathrm{P}(z > z_\alpha) = \alpha$. Olemassa on useita muitakin menetelmiä luotettavuusvälien arvioimiseksi - kaikki aivät vaadi oletusta normaalijakautuneista havainnoista.

R–ohjelman paketit `boot` ja `bootstrap` sisältävät bootstrap menetelmiin liittyviä rutiineja.

Jackknife perustuu myös havaintoaineistosta tehdyn uuden otoksen analysointiin. Tällä kertaa otos on kuitenkin alkuperäistä pienempi, . Esimerkiksi, jos havainnoista poistetaan j:s piste, voidaan havaitun suureen keskiarvon estimaatti laskea kaavasta

$$\bar{x}_{-j} = \frac{1}{n-1}\sum_{i\neq j}^{n} x_i\,,$$

jossa n on alkuperäisten havaintojen lukumäärä. Poistetun alkio arvo voidaan laskea

$$x_j = n\,\bar{x} \; - (n-1)\,\bar{x}_{-j}\,.$$

Tämä voidaan yleistää mielivaltaiseen otoksesta laskettavaan statistiseen tunnuslukuun θ, jonka estimaattia merkitään

$$\hat{\theta} = \phi(x_1, x_2, ..., x_n).$$

ja j:nnen pisteen poiston jälkeen

$$\hat{\theta}_j = \phi(x_1, x_2, ..., x_{j-1}, x_{j+1}, ..., x_n).$$

Keskiarvon tapaan analogisesti j:s *pseudo-arvo* on

$$\hat{\theta^*}_j = n\,\hat{\theta} \; - (n-1)\,\hat{\theta}_{-j}.$$

Jackknife estimaatti parametrille θ saadaan pseudo-arvojen keskiarvona

$$\hat{\theta^*} = \frac{1}{n}\sum_{i=1}^{n} \hat{\theta^*}_i.$$

Estimaatti vaatii siis parametrin laskemista n- kertaa, jolloin kullakin kerralla eri piste on poistettu aineistosta. Estimaatin varianssi on

$$\sigma^2(\hat{\theta^*}) = \sigma^2(\hat{\theta^*}_j)/n = \frac{\sum_{j=1}^{n} (\hat{\theta^*}_j - \hat{\theta^*})^2}{n(n-1)}\,.$$

Tämän perusteella voidaan laskea mielivaltaiset luotettavuusvälit, ja todellinen parametrin arvo osuu välille

$$\hat{\theta^*} \pm t_{\alpha/2,n-1}\;\sigma(\hat{\theta^*})$$

100(1-α) prosentin todennäköisyydellä. Edellä $\mathrm{P}(t_n \geqslant t_{\alpha/2,n-1}) = \alpha$, ja t_n on n–vapausasteen t– jakaumaa noudattava satunnaismuuttuja. Jackknife menetelmä antaa siis mahdollisuuden arvioida määritetyn parametrin virheen pelkkien havaintopisteiden avulla (yksittäisten havaintojen virhearvioita ei tarvita eikä käytetä). Lisäksi itse parametrin estimaatti $\hat{\theta}^*$ on usein parempi kuin koko aineistosta laskettu suora estimaatti – sen bias on pienempi. *R*:ssä Jackknife rutiini löytyy kirjastosta `bootstrap`.

Esimerkki 6.4. Tietystä taivaan osasta on tehty havaintoja toistuvasti kymmenen kertaa ($n = 10$), jolloin kaikkiaan on havaittu $S = 120$ lähdettä. Esim. muuttuvasta kirkkaudesta johtuen $f = 25$ lähdettä on havaittu ainoastaan yhden kerran. Jackknife estimaatti havaittavissa olevien lähteiden kokonaismäärälle on

$$< \hat{\theta}^*_j > = < n\,\hat{\theta} - (n-1)\,\hat{\theta}_{-j} > .$$

Jos oletetaan havaintojen jakautuvan tasaisesti, kussakin otoksessa on 2.5 ainutkertaista lähdettä, eli $< \hat{\theta}_{-j} > = S - f/n$. Lähteiden kokonaismäärän Jackknife estimaatti on toisinsanoen

$$\widehat{S^*} = nS - (n-1)[S - f/n] = S + (\frac{n-1}{n})\,f \;\approx\; S + f\,.$$

Esimerkin tapauksessa

$$\widehat{S^*} = 120 + (\frac{9}{10})25 = 147.5,$$

eli selvästi havaittujen lähteiden lukumäärää suurempi.

Jos sen sijaan kaikki 25 yhden kerran havaittua lähdettä olisivat yhdessä ja samassa otoksessa, olisi estimaatti

$$\widehat{S^*} = 10 \times 120 - (10-1) \times \frac{1 \times 120 + 9 \times 95}{10} = 322.5 \;\; (!) \quad \backslash$$

Esimerkki 6.5. *R*:n `boot`-paketin rutiini `boot` suorittaa bootstrap-otannan, ja palauttaa estimoitavan parametrin arvon sekä sen keskihajonnan. Otantaan on tarjolla eri vaihtoehtoja, mukaanluettuna tasapainotettu otanta. Seuraavassa generoidaan sata lukua normaalijakaumasta, ja lasketaan bootstrap-estimaatit keskihajonnalle sekä pienimmälle arvolle. Rutiinit stdev ja maxi laskevat nämä tunnusluvut vektorin `x` niistä elementeistä, joiden indeksit on annettu indeksivektorissa `ind`. Indeksivektori määrittelee kulloisenkin bootstrap-otoksen.

```
library(boot)
x _ rnorm(100, 0, 1)

stdev _ function(x, ind) {
   sd(x[ind])
}
```

```
print(boot(x, stdev, 10))

maxi _ function(x, ind) {
    max(x[ind])
}
print(boot(x, maxi, 10))
```

Ohjelman tuloksena saadaan:

```
ORDINARY NONPARAMETRIC BOOTSTRAP
Call:
boot(data = x, statistic = stdev, R = 10)

Bootstrap Statistics :
    original     bias    std. error
t1* 1.007713 0.02467861  0.04957076

ORDINARY NONPARAMETRIC BOOTSTRAP
Call:
boot(data = x, statistic = maxi, R = 10)

Bootstrap Statistics :
    original     bias    std. error
t1* 1.970258 -0.0454063   0.1021222
```

Edellä bootsrap otanta toistettiin ainoastaan 10 kertaa (`R=10`) - tavallisesti otoksia kannattaa ottaa vähintää parisataa. Ohjelma palauttaa koko aineistosta lasketun parametrin arvon (`original`), arvioidun biaksen sekä parametrille arvioidun keskihajonnan.

Esimerkki 6.6. Kirjaston `bootstrap` rutiini `boott` laskee mielivaltaisen datasta lasketun funktion luotettavuusvälit bootstrap otannan avulla. Otetaan esimerkiksi vaikkapa tiedostosta sgp_agn eteläisen galaktisen navan alueen kvasaarien värien $B-V$ keskiarvo.

```
library(bootstrap)
SGP _ read.table("sgp_agn.tab", header=TRUE) ;
ind _ which(SGP$ID=="QSO") ;
BV  _ SGP$BV[ind] ;
fun _ function(x) { mean(x) ; }
x   _ bootт(BV, fun) ; \
```

Tulos on seuraavanlainen:

```
$confpoints
     0.001     0.01     0.025     0.05      0.1       0.5
[1,] 0.3654289 0.3750888 0.3799756 0.3824853 0.3872513 0.4020093
     0.90      0.95      0.975     0.99      0.999
[1,] 0.4182367 0.42093   0.4270399 0.4365984 0.4461144
```

Huomaa, että nämä rajat on laskettu ilman, että havaintojen oletetaan olevan normaalijakautuneita. Vastaavat normaaliteorian mukaiset luotettavuusrajat seuraavat otoskeskiarvosta ($\mu = 0.4020093$) ja otoskeskihajonnasta ($\sigma = 0.2461777$). Esimerkiksi piste $P = 0.01$ vastaa muuttujan arvoa $\mu - 2.326\,\sigma/\sqrt{n} \approx 0.3743312$.

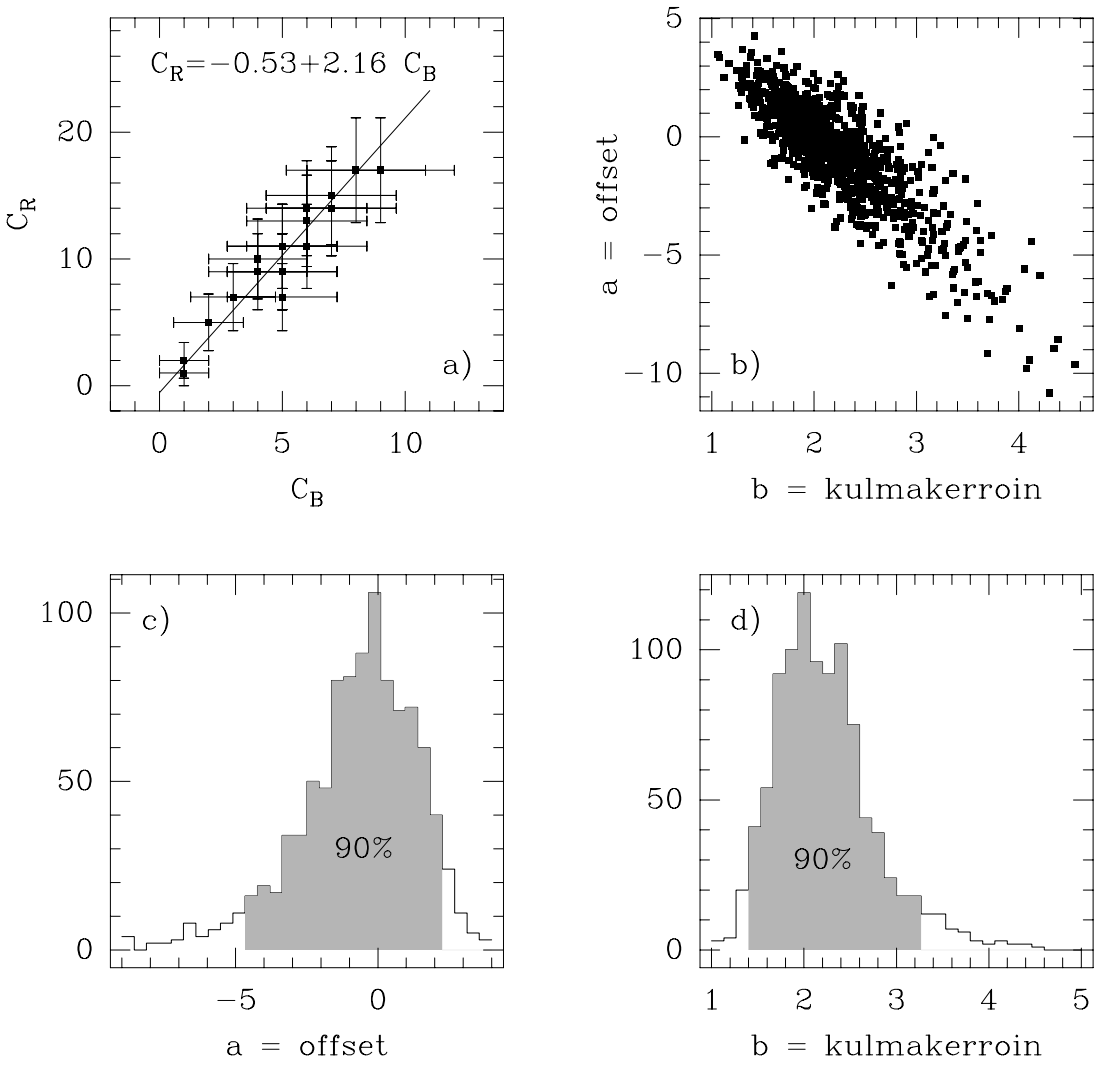
Yleisessä tapauksessa voi olla tarpeen määrittää parametrien virheet tarkemmin muilla menetelmillä. Bootstrap ja Jacknife menetelmät eivät välttämättä käytä tietoa yksittäisten havaintojen epävarmuudesta (itse asiassa ainakin *R*:ssä bootstrap rutiinit *voivat* käyttää myös havaintopisteille annettuja painokertoimia). Jos havaintojen virhejakaumat ovat tiedossa, voidaan virheet arvioida tarkemmin muilla menetelmillä. Lisäksi esim. `boot` rutiini palauttaa ainoastaan yhden virhearvion. Jos parametrin todennäköisyysjakauma on vino, pitäisi arvioida erikseen mahdolliset virheet ylos- ja alaspäin. Jos tulos ilmoitetaan muodossa '5.0 ± 2.2', tämä tarkoittaa implisiittisesti, että virhe on normaalijakautunut ja keskihajonta on 2.2. Normaalijakauman tapauksessa ilmoitetulle välille osuisi 68% mittauksista. Jos jakauma on huomattavan epäsymmetrinen, on syytä laskea rajat todennäköisyystiheysfunktiosta. Etsitään vaikkapa pisteet x_1 ja x_2, joille $P(x_1) = 0.16$ ja $P(x_2) = 0.76$. Näiden väliin jää siten 68% havainnoista ja 68% luotettavuusväli (*confidence interval*) voidaan ilmoittaa seuraavaan tapaan: $5.0^{+3.5}_{-1.9}$. Ensimmäinen vaatimus on, että todellinen todennäköisyystiheysfunktio tunnetaan.

Tavallisesti mallinsovituksesta saatavien parametriarvojen jakaumaa ei suoraan tunneta, sillä jakauma määräytyy monimutkaisella tavalla havaintopisteiden arvioiduista virheistä ja itse mallista. Virheiden jakauma voidaan kuitenkin arvioida **Monte Carlo**-simuloinnilla periaatteessa täysin samoin kuin silloin, kun määritetään minkä tahansa havaintoarvoista riippuvan suureen virhearvioita. Menetelmässä generoidaan uusi havaintopisteiden joukko käyttäen hyväksi todellisia havaintoja ja niiden virhearvioita:

- toista 'riittävän monta kertaa':
    - lisää kuhunkin alkuperäiseen havaintoon satunnainen virhe, joka generoidaan havainnon arvioidusta todennäköisyysjakaumasta
    - sovita malli ja tallenna saadut parametrien arvot
- lue parametriarvojen jakaumasta luotettavuusväli

Tarvittavat satunnaisluvut voi generoida esimerkiksi kertymäfunktiomenetelmällä (katso luku 1.2). Useimmiten havainnot noudattavat kuitenkin riittävällä tarkkuudella esim. normaalijakaumaa tai Poisson-jakaumaa, ja näille löytyvät valmiiksi omat satunnaislukugeneraattorit esim. Matlabista tai IDL:stä.

Esimerkki 6.7. Suoran sovituksen yhteydessä on selvää, ettei kulmakertoimen luotettavuusväli tleensä ole symmetrinen. Suoran jyrkentyessä x-akselista 45^o asteen kulmaan kulmakerroin kasvaa arvosta $k = 0$ arvoon $k = 1$. Seuraavalla 45^o asteen matkalla kulmakerroin kasvaa vastaavasti arvosta $k = 1$ arvoon k=∞. Samoin kulmakertoimen lähestyessä arvoja $k = \pm\infty$ kulkee lineaarisen yhtälön vakiotekijä kohden arvoja $\mp\infty$.



Kuva 6.1. Suoran sovitus pistejoukkoon kun sekä x- että y- suuntaiset virheet on otettu huomioon. Suoran $y = a + bx$ parametrien virhejakaumat on määritetty Monte Carlo menetelmällä (kuva 2). Kuten odotettua, parametrien välinen korrelaatio on selvästi negatiivinen. Alakuvat näyttävät parametriarvojen jakaumat Monte Carlo testissä, sekä parametreille määritetyt 90% luottamusvälit.

6.5 Esimerkki: suoran sovitus

Edellä käsitellyn mallinsovitusongelman pohjalta voidaan nyt tarkastella uudelleen suoransovitusta yleisemmin. Tällä kertaa havaintopisteille on annettu virhearviot sekä $x-$ että $y-$ akselien suhteen. Miten tässä tapauksessa saadaan sovitettavan suoran parametrit ja näiden parametrien virhearviot? Luvun 4.2.1 kaavoista ei ole hyötyä, sillä (1) nyt myös x-suuntaiset virheet on huomioitava ja (2) virheiden ei enää oleteta olevan normaalijakautuneita.

Malli on $y = b + kx$, mutta molemmat muuttujat, x ja y, ovat samanarvoisessa asemassa. Merkitään yksittäisen pisteen todennäköisyysjakaumaa $p_i(x,\ y)$ - kyseessä on yleinen todennäköisyystiheysfunktio, joten myös $x-$ ja $y-$ suuntaisten virheiden välinen korrelaatio on mahdollinen. Pisteen todennäköisyys sovitetun suoran suhteen saadaan marginalisoimalla todennäköisyysjakauma niin, että ainoaksi muuttujaksi jää pisteen etäisyys suorasta so. suoran normaalin suuntaan. Merkitään tätä $p_i^{\perp}(d_i)$, missä d_i on pisteen etäisyys suorasta. Periaatteessa suoran parametrit saadaan maksimoimalla funktiota

$$f(a,b) = \Pi_{i=1}^{N}\, p_i(d_i).$$

Tämän laskemisen tekee työlääksi se, että todennäköisyysjakauman p_i muoto muuttuu aina suoran kulmakertoimen muuttuessa.

Yksinkertaistetaan tehtävää niin, että oletetaan koordinaattiakselien suuntaisten virheiden olevan toisistaan riippumattomia, ja vieläpä normaalijakautuneita (normaalijakaumaoletus helpottaa seuraavia laskuja, mutta jakauman muodolla ei enää ole sellaista periaatteellista merkitystä kuin aiemmissa luvuissa). Merkitään $k = \tan\alpha$, jolloin pisteen etäisyys suorasta on

$$d = (y_i - b)\cos\alpha - x_i \sin\alpha,$$

tai kulmakertoimen avulla kirjoitettuna

$$d = [(y_i - b) - kx_i\,](1 + k^2)^{-1/2}.$$

Seuraavaksi on laskettava pisteiden marginalisoidut todennäköisyysjakaumat suoran normaalin suuntaan. Tässä on apuna tieto, että multinormaalin jakauman kaikki reunajakaumat ovat myös normaaleja. Lisäksi tiedetään normaalijakauman korkeuskäyrien olevan ellipsejä, joten geometrisesta tarkastelusta saadaan reunajakauman keskihajonta normaalin suunnassa,

$$\sigma_i^{\perp} = \{\,[1 + \tan^2\alpha]\,\frac{\sigma_x^2\,\sigma_y^2}{\sigma_y^2 + \sigma_x^2\tan^2\alpha}\,\}^{1/2} = (1 + k^2)\,\frac{\sigma_x^2\,\sigma_y^2}{\sigma_y^2 + k^2\sigma_x^2}\,.$$

Näillä tiedoilla voidaan maksimoitava funktio jo kirjoittaa, sillä yksittäisen pisteen todennäköisyys on

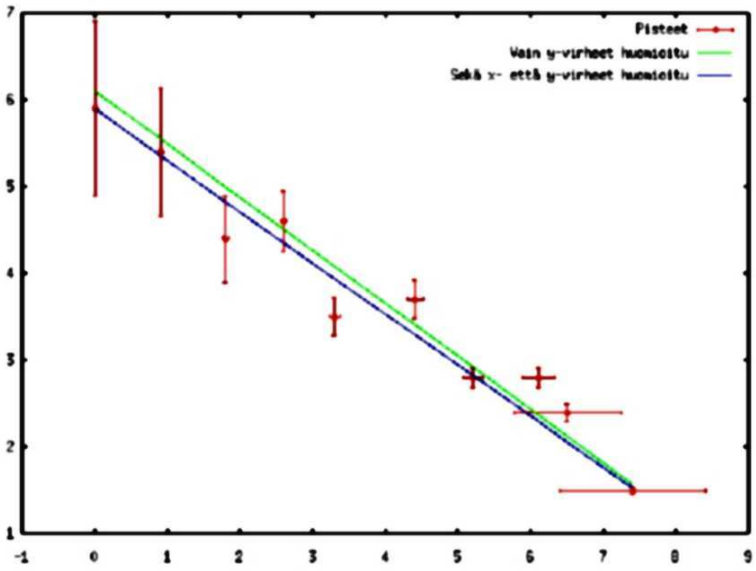
$$p_i = \frac{1}{\sqrt{2\pi}\sigma_i^{\perp}} \exp[\,-\frac{1}{2}\,(\,\frac{d_i}{\sigma_i^{\perp}}\,)^2\,].$$

ML-estimaatit suoran parametreille saadaan millä tahansa optimointialgoritmilla. Todennäköisyysarvojen suuri vaihtelu aiheuttaa kuitenkin helposti numeerisia ongelmia, joten jälleen otetaan funktion logaritmi. Minimoitava funktio on periaatteessa tuttua muotoa,

$$\sum_{i=1}^{N} (\frac{d_i}{\sigma_i})^2,$$

mutta kaavassa esiintyvät d_i ja σ_i ovat molemmat mallin parametrien epälineaarisia funktioita. Parametrien arvot määrätään yleisellä optimointirutiinilla.

Seuraavassa kuvassa vertailun vuoksi tavallinen pienimmän neliösumman suora (ainoastaan y–suuntaiset virheet huomioitu), sekä suora, joka on sovitettu edellisiä kaavoja ja optimointialgoritmia käyttäen.



Kuva 6.2. Kaksi samaan pistejoukkoon sovitettua suoraa. Ensimmäisessä sovituksessa on huomioitu ainoastaan pisteiden y-akselin suuntaiset virheet (ylempi suora). Toisessa sovituksessa myös x-suuntaiset virheet on otettu huomioon (alempi suora).