Monte Carlo -menetelmät bayesilaisessa lineaarisessa regressiossa

Rami Korhonen

LuK-tutkielma
Kesäkuu 2025

Tarkastaja:
Dos. Yury Nikulin

MATEMATIIKAN JA TILASTOTIETEEN LAITOS

TURUN YLIOPISTO
Matematiikan ja tilastotieteen laitos

Rami Korhonen: Monte Carlo -menetelmät bayesilaisessa lineaarisessa regressiossa
LuK-tutkielma, 17 s., 10 liites.
Matematiikka
Kesäkuu 2025

Tässä LuK-tutkielmassa käsitellään bayesilaisen lineaarisen regressioanalyysin ja Monte Carlo -menetelmien perusteita. Tavoitteena on yhdistää klassinen lineaarinen regressioanalyysi bayesilaisen tilastollisen päättelyn periaatteisiin, joissa mallin parametreja tulkitaan satunnaismuuttujina. Tällainen lähestymistapa mahdollistaa mallin parametreihin liittyvän epävarmuuden tarkemman käsittelyn verrattuna perinteisiin piste-estimaatteihin.

Bayesilaisen tilastollisen päättelyn keskeinen etu on se, että sen avulla aiempi tieto ja havaintoaineistosta saatu informaatio voidaan yhdistää muodollisesti posteriorijakaumaksi. Monte Carlo -menetelmät toimivat keskeisessä roolissa posteriorijakaumien numeerisessa arvioinnissa, kun analyyttinen ratkaisu ei ole mahdollinen. Tutkielmassa esitellään miten Monte Carlo -menetelmiä voidaan käyttää bayesilaisen lineaarisen regressiomallin parametrien arviointiin. Näin tutkielma tarjoaa selkeän kokonaiskuvan bayesilaisen tilastotieteen, klassisen lineaarisen regressioanalyysin ja Monte Carlo -menetelmien yhteydestä.

Asiasanat: Bayes-päättely, Monte Carlo -menetelmät, Metropolis-Hastings-algoritmi

Sisällys

1 Johdanto

Lineaarinen regressioanalyysi on yksi keskeisimmistä tilastotieteen ja koneoppimisen menetelmistä. Sen tavoitteena on mallintaa vastemuuttujaa selittävien muuttujien lineaarisena funktiona. Lineaarinen regressioanalyysi on laajalti tutkittu aihe, etenkin klassisen frekventistisen tilastotieteen näkökulmasta, mutta bayesilainen lähestymistapa luo kuitenkin mahdollisuuden tarkastella mallin parametreihin liittyvää epävarmuutta tarkemmin.

Bayesilainen tilastotiede perustuu ehdollisten todennäköisyyksien ja Bayesin kaavan hyödyntämiseen. Sen alku sijoittuu 1700-luvun lopulle [2], mutta mielenkiinto sen tutkimiseen on säilynyt tähän päivään asti, koska useat Bayes-tilastotieteen menetelmät vaativat laskennallista tehokkuutta, minkä nykypäivän teknologia mahdollistaa. Etenkin Monte Carlo -simulaatiomenetelmät ovat yksi keskeisimpiä tällaisia menetelmiä modernissa bayesilaisessa tilastotieteessä.

Tutkielman tavoitteena on esitellä Bayes-tilastotieteen, lineaarisen regressioanalyysin ja Monte Carlo -menetelmien teoriaa, sekä rakentaa näiden aihealueiden välille selkeä ja yhtenäinen yhteys. Keskeisimpänä lähteenä toimii kirja *Bayesian data analysis* (Gelman ym.), josta on tutkielman julkaisuhetkellä tullut päivitetty versio vuonna 2025, mutta tutkielmassa käytetään vanhempaa helmikuun 2021 versiota. Kirjan lisäksi etenkin tarkempaa matemaattista käsittelyä vaativat osiot on täydennetty muitakin lähteitä hyödyntämällä. Käsitteet on käännetty Suomen Tilastoseuran sanaston verkkoversion avulla [12].

Lukijalta oletetaan vähintään perustason ymmärrystä matriisilaskennasta tilastotieteen kontekstissa, todennäköisyyslaskennasta ja tilastollisesta päättelystä. Huomioi myös, että tutkielmassa käytetään termejä *tiheysfunktio* ja *jakauma* toistensa synonyymeina, mutta merkinnällä $p(\cdot)$ tarkoitetaan nimenomaan tiheysfunktiota.

2 Bayesilainen lineaarinen regressio

2.1 Bayesilainen ja frekventistinen tilastollinen päättely

Tilastollisessa päättelyssä pyritään tekemään johtopäätöksiä tuntemattomista ilmiöistä saatavilla olevan havaintoaineiston eli datan perusteella. Päällimmäiseksi eroksi bayesilaisen ja frekventistisen päättelyn välillä voi artikkelin [3] mukaisesti luonnehtia sitä, miten epävarmuutta ja todennäköisyyttä tulkitaan. Frekventistisessä päättelyssä jokin nollahypoteesi oletetaan todeksi ja datan keräämisen ja analysoinnin jälkeen selvitetään uskottavuutta sille, pitääkö nollahypoteesi paikkaansa. Keskeinen suure on *p-arvo*, joka kertoo todennäköisyyden saada vähintään yhtä äärimmäinen havainto kuin havaittu aineisto, kun jokin tällainen nollahypoteesi oletetaan todeksi. Mikäli tämä p-arvo on alle jonkin ennalta määritetyn merkitsevyystason, esimerkiksi 0.05, nollahypoteesi hylätään. Toinen frekventistisille päättelylle ominainen piirre on se, että parametrin arvo oletetaan populaatiotasolla kiinteäksi lukemaksi ja havaintoaineistosta kerätylle parametrin arvon estimaatille voidaan muodostaa jokin *luottamusväli* halutun luottamustason perusteella. Esimerkiksi 95 prosentin luottamusväli tarkoittaa, että jos data kerättäisiin useita kertoja ja jokaiselle aineistolle laskettaisiin luottamusväli, noin 95 % näistä sisältäisi todellisen

populaatiotason parametrin arvon.

Bayesilaisessa päättelyssä parametria ei oleteta kiinteäksi lukemaksi, vaan se mielletään satunnaismuuttujaksi, jota käsitellään todennäköisyyksien avulla. Päättelyssä hyödynnetään parametrista saatavaa ennakkotietoa, eli prioritietoa, joka yhdistetään havaintoaineistoon ja Bayesin kaavan avulla lasketaan *ehdollinen jakauma*, jota kutsutaan *posteriorijakaumaksi* [1]. Tällainen posteriorijakauma ja parametrin tulkinta satunnaismuuttujana mahdollistavat siihen liittyvän epävarmuuden analysoinnin aineiston koosta riippumatta.

2.2 Parametrien estimointi bayesilaisittain

Posteriorijakaumaa laskettaessa pitää tehdä jonkinlainen päätös priorijakaumasta. Tästä syystä Bayes-päättely on altis subjektiivisuudelle, sillä tällainen valinta vaikuttaa posteriorijakaumaan ja sitä kautta johtopäätöksiin parametreista. Priorijakauman valinnassa voidaan hyödyntää esimerkiksi aiempia tutkimustuloksia tai muita ennakkotietoja. Bayes-menetelmiä voidaan hyödyntää toki myös tilanteissa, jossa ennakkotietoja ei ole ja siksi halutaan olla tekemättä vahvoja oletuksia priorijakaumasta. Tällaisiin tilanteisiin sopeutuu *epäinformatiiviset priorit*. Mikäli tarkasteltava parametri on normaalijakauman odotusarvo, epäinformatiiviseksi priorijakaumaksi sopeutuu ([1] s. 64)

$$p(\mu) \propto 1,$$

ja jos tutkittavana on normaalijakauman varianssi, niin

$$p(\sigma^2) \propto \frac{1}{\sigma^2}.$$

Toinen yleisesti käytetty priorijakauma on *konjugaattipriori*, jolla tarkoitetaan sellaista priorijakaumaa, joka kuuluu samaan jakaumaperheeseen kuin posteriorijakauma. Konjugaattipriori yksinkertaistaa posteriorijakauman selvittämistä, mutta se voi olla hankala eksplisiittisesti tietää useiden jakaumien ja parametrien tapauksessa. Lisäksi sen käyttö ei aina ole perusteltua, jos käytettävissä oleva ennakkotieto ei vastaa kyseisen priorijakauman rakennetta.

Määritelmä 1 (*Parametrin posteriorijakauma*)**.** Merkitään tutkittavaa parametrivektoria $\boldsymbol{\theta}$. Olkoon havaintoaineisto D. Tällöin parametrivektorin $\boldsymbol{\theta}$ posteriorijakaumaksi saadaan Bayesin kaavalla ([1] s. 7):

$$p(\boldsymbol{\theta} \mid D) = \frac{p(D \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{P(D)}.$$

Tässä yhteydessä $p(\boldsymbol{\theta})$ on parametrivektorin priorijakauma, $p(D)$ havaintojen reunajakauma ja $p(D \mid \boldsymbol{\theta})$ parametrin *uskottavuusfunktio* ([1] s. 7). Tämä voidaan myös ilmaista

$$p(\boldsymbol{\theta} \mid D) \propto p(D \mid \boldsymbol{\theta})p(\boldsymbol{\theta}),$$

sillä $p(D)$ on täysin parametreista riippumaton normalisointitekijä.

2.3 Yhteys lineaariseen regressioon

2.3.1 Parametrien uskottavuusfunktioista

Perusperiaate Bayes-päättelyssä on se, että priorijakauma yhdistetään uskottavuusfunktion kautta posteriorijakaumaksi. Lineaarisen regression tapauksessa havaintoaineisto D koostuu vastemuuttujista $\mathbf{y}$ ja selittävistä muuttujista $\mathbf{X}$. Tästä syystä mallia muodostaessa selittävät muuttujat $\mathbf{X}$ ovat tunnettuja, jonka takia parametrien päättely ei kohdistu selittäviin muuttujiin $\mathbf{X}$ ([1] s. 354) vaan yleisesti parametreille pätee

$$p(\boldsymbol{\theta} \mid D) = p(\boldsymbol{\theta} \mid \mathbf{y}, \mathbf{X}) = \frac{p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{y}, \mathbf{X})} \propto p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta})p(\boldsymbol{\theta}), \tag{1}$$

eli esimerkiksi myöhemmin esiteltävää odotusarvoa $\boldsymbol{\mu} = \mathbf{X}\boldsymbol{\beta}$ tutkiessa tilastollinen päättely kohdistuu nimenomaan parametrivektoriin $\boldsymbol{\beta}$. Lineaarisen mallin parametrien tapauksessa uskottavuusfunktio on puolestaan

$$p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\beta}, \sigma^2) = (2\pi\sigma^2)^{-\frac{n}{2}} e^{-\frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})} \tag{2}$$

seuraten multinormaalijakauman tiheysfunktion määritelmästä.

2.3.2 Lineaaristen mallien perusperiaatteet

Klassisen lineaarisen regression tapauksessa oletetaan, että tutkittava vastemuuttuja $\mathbf{y}$ on normaalijakautunut. Aineisto jossa on n kappaletta havaintoja ja $k = p - 1$ kappaletta selittäviä muuttujia, voidaan ilmaista lineaarisena regressiomallina

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{3}$$

jossa $\mathbf{y} \mid \mathbf{X} \sim N(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I}_n)$, eli $\boldsymbol{\epsilon} \mid \mathbf{X} \sim N(\mathbf{0}, \sigma^2\mathbf{I}_n)$. Tässä $\mathbf{X}$ on $n \times p$-matriisi kunkin havainnon selittävistä muuttujista, jossa ensimmäinen sarake on pelkästään ykkösiä vakiotermin takia. $\boldsymbol{\beta}$ on $p \times 1$-vektori, joka sisältää vakiotermin (engl. intercept) ja kunkin selittävän muuttujan kerroinparametrin. $\mathbf{y}$ on $n \times 1$-vektori, joka sisältää jokaisen havainnon toteutuneen vastemuuttujan arvon. $\boldsymbol{\epsilon}$ on $n \times 1$-vektori, joka sisältää kunkin havainnon virhetermin, eli todellisen arvon ja ennusteen erotuksen. $\mathbf{I}_n$ on puolestaan $n \times n$ -identiteettimatriisi. Vaihtoehtoisesti lineaarisen mallin voi ilmaista myös

$$y_i = \mathbf{x_i}\boldsymbol{\beta} + \epsilon_i, \qquad i = 1, .., n \tag{4}$$

jossa y_i on havainnon i vastemuuttujan arvo, $\mathbf{x_i}$ on $1 \times p$ vektori jossa on vakio 1 ja havainnon i selittävien muuttujien arvot ja ϵ_i on havainnon i virhetermi. Avattuna siis

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik} + \epsilon_i.$$

Vastemuuttujan normaalisuuden lisäksi oletetaan, että mallissa virhetermit ovat riippumattomia toisistaan ja vastemuuttujasta. Lisäksi oletetaan, että virhetermeillä on yhtä suuri varianssi. Nämä oletukset näkee myös selkeämmin avaamalla kovarianssimatriisi

$$\sigma^2\mathbf{I_n} = \begin{bmatrix} \sigma^2 & 0 & \cdots & 0 \\ 0 & \sigma^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma^2 \end{bmatrix}, \tag{5}$$

jossa kovarianssit ovat siis 0 ja varianssi on vakio.

Se, että jokin näistä ehdoista ei toteudu täydellisesti, ei sinänsä tee lineaarisesta mallista täysin käyttökelvotonta, mutta mallin muodostus perustuu näihin oletuksiin.

Esitellään seuraavaksi yleisiä estimaatteja parametreille.

Lause 1. *$\boldsymbol{\beta}$:n suurimman uskottavuuden estimaatti on*

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}.$$

Todistus. Otetaan luonnollinen logaritmi kaavan (2) uskottavuusfunktiosta, jolloin saadaan log-uskottavuus:

$$-\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}).$$

Tämä maksimoituu $\boldsymbol{\beta}$:n suhteen, kun $(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$, eli virhetermien neliösumma minimoituu. Tämä saadaan muotoon

$$(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) = \mathbf{y}^T\mathbf{y} - 2\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta},$$

jonka minimi on gradientin nollakohta

$$\begin{aligned}
&\frac{d}{d\boldsymbol{\beta}}(\mathbf{y}^T\mathbf{y} - 2\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}) = 0 \\
&\iff -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} = 0 \\
&\iff \mathbf{X}^T\mathbf{X}\boldsymbol{\beta} = \mathbf{X}^T\mathbf{y} \\
&\iff \boldsymbol{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y},
\end{aligned}$$

koska toinen derivaatta $2\mathbf{X}^T\mathbf{X}$ on positiividefiniitti. □

Lause 2. *Olkoon $\mathbf{y} \sim N(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I}_n)$ ja $\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. Tällöin*

$$\hat{\boldsymbol{\beta}} \sim N(\boldsymbol{\beta}, \sigma^2(\mathbf{X}'\mathbf{X})^{-1}).$$

Todistus. Tämä todistus on suoraan lähteen [4] sivulta 28. Odotusarvoksi saadaan

$$\begin{aligned}
E[\hat{\boldsymbol{\beta}}] &= E[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y})] \\
&= (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'E[\mathbf{y}] \\
&= \boldsymbol{\beta}
\end{aligned}$$

ja kovarianssiksi

$$\begin{aligned}
\mathrm{cov}(\hat{\boldsymbol{\beta}}) &= \mathrm{cov}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}) \\
&= (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathrm{cov}(\mathbf{y})\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1} \\
&= \sigma^2(\mathbf{X}'\mathbf{X})^{-1}.
\end{aligned}$$

$\hat{\boldsymbol{\beta}}$:n normaalisuus seuraa siitä, että se on multinormaalisen $\mathbf{y}$:n täysiasteinen lineaarinen muunnos. □

Seuraus 1. *Koska* $E[\hat{\boldsymbol{\beta}}] = \boldsymbol{\beta}$*, on* $\hat{\boldsymbol{\beta}}$ *harhaton estimaattori.*

Varianssiparametrin tapauksessa suurimman uskottavuuden estimaatti ja harhaton estimaattori eivät ole samoja. Yleisesti käytetään kuitenkin harhatonta estimaattoria.

Lause 3. *Varianssiparametrin* σ^2 *harhaton estimaattori on*

$$S^2 = \frac{1}{n-p}(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})^T(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}).$$

Todistus. Käyttämällä projektiomatriisia saadaan

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \boldsymbol{P}\mathbf{y},$$

jossa $\boldsymbol{P} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$, jolloin

$$\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{y} - \boldsymbol{P}\mathbf{y} = (\mathbf{I} - \boldsymbol{P})\mathbf{y}.$$

Nyt, koska $\boldsymbol{P}\mathbf{X} = \mathbf{X}$, saadaan

$$(\mathbf{I} - \boldsymbol{P})\mathbf{y} = (\mathbf{I} - \boldsymbol{P})(\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}) = (\mathbf{I} - \boldsymbol{P})\boldsymbol{\epsilon}.$$

Tällöin

$$(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})^T(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}) = \boldsymbol{\epsilon}^T(\mathbf{I} - \boldsymbol{P})^T(\mathbf{I} - \boldsymbol{P})\boldsymbol{\epsilon} = \boldsymbol{\epsilon}^T(\mathbf{I} - \boldsymbol{P})\boldsymbol{\epsilon},$$

josta saadaan

$$E[S^2] = \frac{1}{n-p}E[(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})^T(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})] = \frac{\sigma^2\mathrm{tr}(\mathbf{I} - \boldsymbol{P})}{n-p} = \frac{\sigma^2(n-p)}{n-p} = \sigma^2.$$

Tässä toinen yhtäsuuruus seuraa ominaisuudesta $E[\boldsymbol{\epsilon}^T A\boldsymbol{\epsilon}] = \sigma^2\mathrm{tr}(A)$, kun on oletettu $\boldsymbol{\epsilon} \sim N(0, \mathbf{I}_n\sigma^2)$. □

Esitetään vielä vertailuksi parametrille $\boldsymbol{\beta}$ bayesilainen posteriorijakauma esimerkin kautta. Käytetään priorijakaumana epäinformatiivista prioria.

Esimerkki 1. Selvitetään $\boldsymbol{\beta}$-parametrin posteriorijakauma. Oletetaan $p(\boldsymbol{\beta}) \propto 1$. Esimerkin yksinkertaistamiseksi oletetaan lisäksi, että σ^2 tunnettu, joka jätetään merkitsemättä merkintöjen yksinkertaistamiseksi. Posteriorijakaumaksi tulee tällöin

$$p(\boldsymbol{\beta} \mid D) \propto p(D \mid \boldsymbol{\beta})p(\boldsymbol{\beta}) \propto p(D \mid \boldsymbol{\beta}),$$

jossa

$$p(D \mid \boldsymbol{\beta}) \propto e^{-\frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})} = e^{-\frac{1}{2\sigma^2}(\mathbf{y}^T\mathbf{y}-2\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y}+\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta})}.$$

Sijoitetaan tähän lauseen 1 todistuksessa käytetty $\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}^T\mathbf{y}$ ja täydennetään neliöksi lisäämällä ja vähentämällä $\hat{\boldsymbol{\beta}}^T\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}}$, jolloin saadaan

$$\begin{aligned}
e^{-\frac{1}{2\sigma^2}(\mathbf{y}^T\mathbf{y}-2\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y}+\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta})} &= e^{-\frac{1}{2\sigma^2}(\mathbf{y}^T\mathbf{y}-2\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}}+\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}+\hat{\boldsymbol{\beta}}^T\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}}-\hat{\boldsymbol{\beta}}^T\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}})} \\
&= e^{-\frac{1}{2\sigma^2}((\boldsymbol{\beta}-\hat{\boldsymbol{\beta}})\mathbf{X}^T\mathbf{X}(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}})+\mathbf{y}^T\mathbf{y}-\hat{\boldsymbol{\beta}}^T\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}}))} \\
&\propto e^{-\frac{1}{2\sigma^2}((\boldsymbol{\beta}-\hat{\boldsymbol{\beta}})\mathbf{X}^T\mathbf{X}(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}}))},
\end{aligned}$$

joka on multinormaalijakauman $N(\hat{\boldsymbol{\beta}}, \sigma^2(\mathbf{X}^T\mathbf{X})^{-1})$ tiheysfunktion normalisoimaton muoto. Tästä seuraa, että

$$\boldsymbol{\beta} \mid D \sim N(\hat{\boldsymbol{\beta}}, \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}).$$

Mikäli varianssi on tuntematon, kuten se usein käytännön toteutuksissa on, voidaan molempien parametrien yhteisposteriori selvittää niin, että ensiksi selvitetään $\boldsymbol{\beta} \mid \sigma^2$ posteriori kuten yllä ja sen jälkeen selvitetään σ^2 reunajakauman posteriori. ([1] s.355)

3 Monte Carlo -menetelmät Bayes-päättelyssä

3.1 Monte Carlo -menetelmien käytöstä

Monte Carlo -menetelmillä tarkoitetaan laskentamenetelmiä, joissa analyyttisen ratkaisun sijaan pyritään numeerisen simuloinnin ja satunnaislukujen generoinnin avulla saavuttamaan approksimatiivisia ratkaisuja. ([5] s. 1).

Analyyttinen ratkaisu posteriorijakaumalle voidaan saavuttaa tietyissä tapauksissa melko suoraviivaisesti, esimerkiksi silloin, kun priorijakauma on konjugaattipriori. Usein tulee kuitenkin tilanteita, joissa posteriorijakauman laskeminen täysin analyyttisesti osoittautuu hankalaksi normalisointitekijän $p(D) = p(\mathbf{X}, \mathbf{y})$ selvittämisen takia. Jatkuvan parametrin tapauksessa normalisointitekijänä toimivan datan reunajakauma on

$$p(D) = \int_{\boldsymbol{\theta}} p(D, \boldsymbol{\theta}) d\boldsymbol{\theta} = \int_{\boldsymbol{\theta}} p(D \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) \boldsymbol{d\theta}, \tag{6}$$

jossa siis integroidaan kaikkien mahdollisten parametrivektorin $\boldsymbol{\theta}$ arvojen yli. Lineaarisen mallin parametrivektorin $\boldsymbol{\beta}$ tapauksessa integraalin ulottuvuus määräytyy selittävien muuttujien määrän mukaan. Tästä syystä, mikäli selittävien muuttujien määrä on suuri, analyyttisen ratkaisun selvittäminen voi olla vaikeaa. Samasta syystä tarkastelun kohteena ovat erityisesti Markov-ketju Monte Carlo (MCMC) -menetelmät, sillä ne sopeutuvat tavallisia Monte Carlo -menetelmiä paremmin korkeamman ulottuvuuden malleihin etenkin hierarkkisissa malleissa, joissa parametrien välillä voi olla riippuvuussuhteita ([1] s.262).

3.2 Markov-ketjuista ja niiden simuloinnista

Markov-ketjut voidaan karkeasti jakaa diskreetti- ja jatkuva-aikaisiin sekä numeroituva- ja ylinumeroituvatilaisiin Markov-ketjuihin. Tässä *tila* tarkoittaa stokastisen prosessin tila-avaruuden eli mahdollisten arvojen joukon arvoa jollain hetkellä. Myöhemmin esiteltäviä MCMC-menetelmiä voidaan sinänsä soveltaa kaikkiin näistä, mutta koska parametrien päivitystä tehdään diskreetissä ajassa ja normaalijakauman parametrit ovat reaalilukuja, keskitytään diskreettiaikaisiin ja ylinumeroituvatilaisiin Markov-ketjuihin. Esitetään muutama tärkeä näitä koskeva määritelmä.

Määritelmä 2 (*Diskreettiaikainen Markov-ketju*)**.** Olkoon $\{X_1, ..., X_n\}$ jono satunnaismuuttujia tila-avaruudessa $\mathcal{X}$. Olkoon A jokin tila-avaruuden $\mathcal{X}$ mitallinen osajoukko. Jono on *Markov-ketju*, jos kaikilla $i \in \{1, ..., n\}$ ja kaikilla A:

$$\mathbb{P}(X_i \in A \mid X_{i-1}, ..., X_1) = \mathbb{P}(X_i \in A \mid X_{i-1}).$$

([6] s.4)

Määritelmä 3. Olkoon $(X, \mathcal{X})$ ja $(Y, \mathcal{Y})$ mitallisia avaruuksia. Siirtymäydin K X:stä Y:hyn on funktio

$$K : X \times \mathcal{Y} \to [0, 1],$$

jolla:
(1) Jokaiselle kiinnitetylle $x \in X$ funktio $A \mapsto K(x, A)$ on todennäköisyysmitta avaruudessa $(Y, \mathcal{Y})$.
(2) Jokaiselle kiinnitetylle $A \in \mathcal{Y}$, funktio $x \mapsto K(x, A)$ on mitallinen joukossa X. Tässä siis

$$\mathbb{P}(X_i \in A \mid X_{i-1}) = \int_A K(X_{i-1}, x^*)dx^*.$$

([6] s.6)

Määritelmä 4. Markov-ketju täyttää *tilaparikohtaisen tasapainoehdon*, jos on olemassa funktio f, jolla

$$K(y, x)f(y) = K(x, y)f(x)$$

kaikilla (x, y) ([7] s. 235).

Markov-ketjuihin perustuvia MCMC-menetelmiä hyödynnetään Bayes-päättelyssä niin, että otantaa suoritetaan iteratiivisesti askel askeleelta. Iteraatiot lähestyvät todellista posteriorijakaumaa $p(\boldsymbol{\theta} \mid D)$ muodostaen Markov-ketjun ([1] s. 275). Posteriorijakaumaa ei siis selvitetä eksplisiittisesti, vaan pyritään vain suorittamaan otantaa sieltä. Keskeisiä tällaisia menetelmiä ovat esimerkiksi Hastings-algoritmi, Metropolis-Hastings-algoritmi ja Gibbsin otanta.

3.3 Metropolis-Hastings-algoritmi

Metropolis-Hastings-algoritmissa tulee esille käsitteet *kohdejakauma*, *ehdokasjakauma* ja *hyväksymistodennäköisyys*. Kohdejakaumalla tarkoitetaan jakaumaa josta pyritään suorittamaan otantaa, joka on tässä tapauksessa posteriorijakauma $p(\boldsymbol{\theta} \mid D)$. Ehdokasjakaumalla tarkoitetaan sellaista jakaumaa $q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{t-1})$ joka generoi uusia arvoja $\boldsymbol{\theta}^*$. Hyväksymistodennäköisyys $\mathcal{A}(\boldsymbol{\theta}^{t-1}, \boldsymbol{\theta}^*)$ kertoo todennäköisyyden uuden ehdotuksen $\boldsymbol{\theta}^*$ hyväksymiselle. ([11] s. 15 ja [1] s. 278).

Ehdokasjakauman valintaan on useita eri vaihtoehtoja, kuten normaalijakauma, tasajakauma, Cauchyn jakauma, eksponenttijakauma ja t-jakauma ([9] s. 43-44). Sen valinta voi olla hankalaa, ja pääsääntöisesti mitä lähempänä ehdokasjakauma on kohdejakaumaa, sitä tehokkaammin algoritmi toimii. Tärkeä piirre kuitenkin ehdokasjakaumalle on se, että siitä on helppo suorittaa otantaa, ja hyväksymistodennäköisyys on helppo laskea ([1]s. 280). Lisäksi on suotavaa, että ehdokasjakauma mahdollistaa koko posteriorijakauman kantajan tutkimisen (lisätietoa alaluvussa 3.5).
Hyväksymistodennäköisyys määritellään ([11] s.16)

$$\mathcal{A}(\boldsymbol{\theta}^{t-1}, \boldsymbol{\theta}^*) = \min\{1, \frac{p(\boldsymbol{\theta}^* \mid D)q(\boldsymbol{\theta}^{t-1} \mid \boldsymbol{\theta}^*)}{p(\boldsymbol{\theta}^{t-1} \mid D)q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{t-1})}\}, \tag{7}$$

jonka selvittäminen onnistuu tietämättä normalisointitekijää, sillä se supistuu pois

$$\frac{p(\boldsymbol{\theta}^* \mid D)}{p(\boldsymbol{\theta}^{t-1} \mid D)} = \frac{p(D \mid \boldsymbol{\theta}^*)p(\boldsymbol{\theta}^*)/p(D)}{p(D \mid \boldsymbol{\theta}^{t-1})p(\boldsymbol{\theta}^{t-1})/p(D)} = \frac{p(D \mid \boldsymbol{\theta}^*)p(\boldsymbol{\theta}^*)}{p(D \mid \boldsymbol{\theta}^{t-1})p(\boldsymbol{\theta}^{t-1})}. \tag{8}$$

Algoritmi etenee siis seuraavalla tavalla ([11] s. 16 ja [1] s. 278):

Metropolis-Hastings-algoritmi

Askel	
1	Tehdään otanta jostain aloitusjakaumasta $p_0(\boldsymbol{\theta})$ ja käytetään sitä alkuarvona $\boldsymbol{\theta}^0$.
2.1	Suoritetaan otanta ehdokasjakaumasta $\boldsymbol{\theta}^* \sim q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{t-1})$.
2.2	Lasketaan $\mathcal{A}(\boldsymbol{\theta}^{t-1}, \boldsymbol{\theta}^*)$.
2.3	Arvotaan luku u väliltä $[0, 1]$, suorittamalla otanta $U(0, 1)$-jakaumasta.
2.4	Jos $u < \mathcal{A}(\boldsymbol{\theta}^{t-1}, \boldsymbol{\theta}^*) \rightarrow$ asetetaan $\boldsymbol{\theta}^t = \boldsymbol{\theta}^*$. Jos $u \geq \mathcal{A}(\boldsymbol{\theta}^{t-1}, \boldsymbol{\theta}^*) \rightarrow$ asetetaan $\boldsymbol{\theta}^t = \boldsymbol{\theta}^{t-1}$. Palataan askeleeseen 2.1.

Päättely perustuu ajatukseen, että tarpeeksi suurella t:n arvolla $\boldsymbol{\theta}^t \sim \boldsymbol{\theta} \mid D$. Tämän takia etenkin alkupään iteroidut arvot eivät kuvasta todellisen posteriorijakauman käyttäytymistä, jonka takia ne yleensä hylätään ja poistetaan tarkastelusta. Esimerkiksi jos iteraatioita suoritetaan 600, voidaan ensimmäiset 300 iteraatioita poistaa käsittelystä ja tarkastella vain jälkimmäistä 300 iteraatiota ([1] s. 282). Näitä poistettavia alkupään iteraatioita kutsutaan *sisäänajojaksoksi* (engl. burn-in period).

Aiemmin mainitut Metropolis-algoritmi ja Gibbsin otanta voidaan nähdä Metropolis-Hastings-algoritmin erikoistapauksina. Metropolis-algoritmi vastaa hyvin paljon Metropolis-Hastings-algoritmia, mutta siinä oletetaan ehdokasjakauman olevan symmetrinen, jolloin ehto $q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{t-1}) = q(\boldsymbol{\theta}^{t-1} \mid \boldsymbol{\theta}^*)$ toteutuu ([1] s. 278). Tällöin

$$\frac{p(\boldsymbol{\theta}^* \mid D)q(\boldsymbol{\theta}^{t-1} \mid \boldsymbol{\theta}^*)}{p(\boldsymbol{\theta}^{t-1} \mid D)q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{t-1})} = \frac{p(\boldsymbol{\theta}^* \mid D)}{p(\boldsymbol{\theta}^{t-1} \mid D)}$$

jolloin hyväksymistodennäköisyys on yksinkertaisempaa muotoa

$$\mathcal{A}(\boldsymbol{\theta}^{t-1}, \boldsymbol{\theta}^*) = \min\{1, \frac{p(\boldsymbol{\theta}^* \mid D)}{p(\boldsymbol{\theta}^{t-1} \mid D)}\}. \tag{9}$$

Muuten Metropolis-algoritmi etenee samalla tavalla kuin Metropolis-Hastings.

3.4 Gibbsin otanta

Gibbsin otannassa parametrivektori $\boldsymbol{\theta}$ jaetaan k:hon osavektoriin $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_k)$. Jokaisella iteraatioaskeleella käydään jokainen osavektori läpi, niin että jokaisen osavektorin otanta suoritetaan ehdollistettuna muille osavektoreille. Olkoon iteraation t osavektori j $\boldsymbol{\theta}_j^t$. Selvitetään siis ([1] s. 277)

$$p(\boldsymbol{\theta}_j^t \mid \boldsymbol{\theta}_{-j}^{t-1}, D), \tag{10}$$

jossa $\boldsymbol{\theta}_{-j}^{t-1}$ on kaikkien osavektoreiden, paitsi j:n arvo kyseisellä hetkellä, eli

$$\boldsymbol{\theta}_{-j}^{t-1} = (\boldsymbol{\theta}_1^t, ..., \boldsymbol{\theta}_{j-1}^t, \boldsymbol{\theta}_{j+1}^{t-1}, ..., \boldsymbol{\theta}_k^{t-1}).$$

Tarkastellessa tätä tarkemmin, Gibbsin otanta voidaan nähdä Metropolis-Hastings-algoritmin erikoistapauksena. Jokaisella iteraatiolla t on k askelta, jossa askel j nähdään vektorin $\boldsymbol{\theta}_j$ päivityksenä ehdollistettuna kaikille muille osavektoreille. Tällöin ehdokasjakauma liikkuu vain osavektorilla j, jolloin saadaan ([1] s.281)

$$q_{j,t}^{\text{Gibbs}}(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{t-1}) = \begin{cases} p(\boldsymbol{\theta}_j^* \mid \boldsymbol{\theta}_{-j}^{t-1}, y), \text{ kun } \boldsymbol{\theta}_{-j}^* = \boldsymbol{\theta}_{-j}^{t-1} \\ 0, \text{ muulloin.} \end{cases} \tag{11}$$

Tällöin hyväksymistodennäköisyydeksi saadaan

$$\begin{aligned} \mathcal{A}^{\text{Gibbs}}(\boldsymbol{\theta}^{t-1}, \boldsymbol{\theta}^*) &= \min\{1, \frac{p(\boldsymbol{\theta}^* \mid D) q^{\text{Gibbs}}(\boldsymbol{\theta}^{t-1} \mid \boldsymbol{\theta}^*)}{p(\boldsymbol{\theta}^{t-1} \mid D) q^{Gibbs}(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{t-1})}\} \\ &= \min\{1, \frac{p(\boldsymbol{\theta}^* \mid D) p(\boldsymbol{\theta}_j^{t-1} \mid \boldsymbol{\theta}_{-j}^{t-1}, D)}{p(\boldsymbol{\theta}^{t-1} \mid D) p(\boldsymbol{\theta}_j^* \mid \boldsymbol{\theta}_{-j}^{t-1}, D)}\} \\ &= \min\{1, \frac{p(\boldsymbol{\theta}_{-j}^{t-1} \mid D)}{p(\boldsymbol{\theta}_{-j}^{t-1} \mid D)}\} = 1. \end{aligned}$$

Huomataan, että vaikka Gibbsin otantaa ei ole alun perin kehitetty Metropolis-Hastings-algoritmin mukaiseksi, se voidaan nähdä algoritmin erikoistapauksena, jossa jokainen ehdokas hyväksytään automaattisesti.

3.5 Suppenemisdiagnostiikka

Keskeinen asia MCMC-menetelmien toimivuuden kannalta on se, että tarkasteltava Markov-ketju suppenee kohti jotain invarianttia jakaumaa, joka on nyt posteriorijakauma $p(\boldsymbol{\theta} \mid D)$. Esitellään ehtoja tämän toteutumiselle.

Lause 4. *Oletetaan, että siirtymäydin K täyttää määritelmän 4 mukaisen tilaparikohtaisen tasapainoehdon funktiolla f. Tällöin f on invariantti tiheysfunktio Markov-ketjulle ja Markov-ketju on kääntyvä.*

Todistus. Katso ([7] s. 235) □

Metropolis-Hastings-algoritmin muodostama siirtymäydin on siis määritelmän 3 mukaisesti

$$K(\boldsymbol{\theta}^{t-1}, \boldsymbol{\theta}^*) = \mathcal{A}(\boldsymbol{\theta}^{t-1}, \boldsymbol{\theta}^*) q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{t-1}) + (1 - \int \mathcal{A}(\boldsymbol{\theta}^{t-1}, \boldsymbol{\theta}^*) q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{t-1}) d\boldsymbol{\theta}^*) \delta_{\boldsymbol{\theta}^{t-1}}(\boldsymbol{\theta}^*), \tag{12}$$

jossa δ on Diracin mitta. Yhdistämällä tämä jakaumaan $p(\boldsymbol{\theta} \mid D)$ huomataan, että $K(\boldsymbol{\theta}^{t-1}, \boldsymbol{\theta}^t) p(\boldsymbol{\theta}^{t-1} \mid D) = K(\boldsymbol{\theta}^t, \boldsymbol{\theta}^{t-1}) p(\boldsymbol{\theta}^t \mid D)$, ([7] s.235) kuitenkin edellyttäen, että kohdejakauman kantaja sisältyy kokonaan ehdokasjakauman kantajaan. Nyt koska Markov-ketju täyttää tilaparikohtaisen tasapainoehdon posteriorijakaumalla $p(\boldsymbol{\theta} \mid D)$, on posteriorijakauma lauseen 4 nojalla invariantti tiheys Markov-ketjulle.

Tämän lisäksi on tärkeää, että algoritmin muodostama Markov-ketju on *ergodinen*, eli keskiarvosuppeneva. Tällä tarkoitetaan sitä, että halutaan varmistaa, että Markov-ketju todellakin suppenee kohti tällaista invarianttia jakaumaa. Lähteen [10] lauseen 1 nojalla tämä toteutuu, kun ketju on *pelkistymätön* (engl. irreducible), *positiivisesti palautuva* (engl. positive recurrent) ja *jaksoton* (engl. aperiodic). Ketjun pelkistymättömyys tarkoittaa, että on mahdollisuus että mistä vaan tilasta voidaan päästä mihin vaan tilaan jossa $p(\boldsymbol{\theta} \mid D) > 0$. Tämä toteutuu, kun ehdokasjakauma q on sellainen, että sen tiheysfunktion arvo on suurempaa kuin 0 kohdejakauman kantajassa. Ketjun pelkistämättömyys pitää siis huolen, että ei synny tilannetta jossa ketjulla on todennäköisyys 0 päätyä tilaan, joka todellisuudessa on mahdollinen posteriorijakaumalle. Mikäli ketju on pelkistymätön, se on positiivisesti palautuva jos ketju palaa johonkin sen osajoukkoon äärellisessä ajassa todennäköisyydellä 1. Tämän ehdon tarkoitus on pitää huoli siitä, että ketju ei "karkaa"esimerkiksi äärettömään. Jaksottomuus puolestaan tarkoittaa sitä, että ketju ei jumiudu johonkin deterministiseen sykliin joka toistuu. Metropolis-Hastings-algoritmin ergodinisuuden tarkempi käsittely löytyy lähteen [7] luvuista 4 ja 6 sekä lähteen [8] luvusta 13.

Kuten aiemmin mainittu, Metropolis-Hastings-algoritmi toimii tehokkaammin, mitä osuvampi ehdokasjakauman valinta on. Kuitenkin huolimatta siitä, kuinka hyvin ehdokasjakauma on valittu, algoritmin suppenemisen teoria nojautuu pitkälti asymptotiikkaan. Tämän takia käytännön toteutuksissa on syytä erikseen tarkastella onko Markov-ketju supennut. Ketjun suppenemista voidaan tutkia muutamilla eri tavoilla. Pelkästään algoritmin muodostaman ketjun graafinen tarkastelu voi paljastaa, mikäli ketju ei ole supennut, mutta suppenemisen tarkasteluun on kehitetty myös suureita. Keskeinen tällainen on Gelman-Rubin tilasto ([1] s.285)

$$\hat{R} = \sqrt{\frac{\frac{n_s-1}{n_s}W + \frac{1}{n_s}B}{W}}, \tag{13}$$

joka perustuu siihen, että simulaatiot tehdään useamman kerran. Tällöin saadaan useampi ketju, jolle lasketaan ketjujen sisäisten otosvarianssien keskiarvo W ja ketjujen keskiarvojen varianssi B. Näistä otetaan painotettu keskiarvo, jonka voi nähdä eräänlaisena parametrin posteriorivarianssin estimaattina, joka jaetaan W:llä. Tässä siis jokaisesta ketjusta poistetaan sisäänajojakso, jonka jälkeen ketju jaetaan vielä kertaalleen kahtia alku- ja loppupuoliskoon. Tällöin ketjuja on kaksinkertainen määrä alkuperäisten simulaatioiden määrään verrattuna. n_s on havaintojen lukumäärä yksittäisessä simulaatioketjussa tämän hajoittamisen jälkeen ja se on kaikille ketjuille sama. Huomaa, että nämä lasketaan yksittäisille parametreille, eli jos parametrivektori $\boldsymbol{\theta}$ sisältää useamman parametrin, niin kaikille yksittäisille skalaareille parametreille lasketaan oma $\hat{R}$. Merkitään θ_{ij}:llä ketjun $j \in \{1, ..., m\}$ havaintoa $i \in \{1, ..., n_s\}$ jostain yksittäisestä skalaariparametrista θ, jolloin saadaan ([1] s.284)

$$B = \frac{n_s}{m-1}\sum_{j=1}^{m}(\bar{\theta}_j - \bar{\theta}_{ij})^2 \tag{14}$$

ja

$$W = \frac{1}{m}\sum_{j=1}^{m}(\frac{1}{n_s - 1}\sum_{i=1}^{n_s}(\theta_{ij} - \bar{\theta}_j)^2), \tag{15}$$

jossa $\bar{\theta}_j = \frac{1}{n_s}\sum_{i=1}^{n}\theta_{ij}$ on ketjun j havaintojen keskiarvo ja $\bar{\theta}_{ij} = \frac{1}{m}\sum_{i=1}^{m}\bar{\theta}j$ on keskiarvo ketjujen keskiarvoista. Tässä on hyvä huomata asymptoottisiin ominaisuksiin liittyen, että

$$\lim_{n_s \to \infty} \hat{R} = 1,$$

ja $\hat{R}$:sta voidaankin ajatella, että mitä lähemmäksi päästään arvoa 1, sen parempi. "Hyvän" $\hat{R}$:n arvon määrittäminen on tilannekohtaista, mutta nyrkkisääntönä pidetään kuitenkin rajaa $\hat{R} < 1.1$ ([1] s. 287).

Toinen käytetty mittari on tehollinen otoskoko (engl. effective sample size). Tämän tarkoitus on tarkastella sitä, kuinka riippuvaisia havainnot ovat toisistaan eri viiveillä t. Se määritellään ([1] s. 286)

$$n_e = \frac{mn_s}{1 + 2\sum_{t=1}^{\infty}\rho_t}, \tag{16}$$

jossa ρ_t on yksittäisen skalaariparametrin muodostaman jonon autokorrelaatio viiveellä t. Estimaattina tälle käytetään ([1] s. 286)

$$\hat{\rho}_t = 1 - \frac{V_t}{2(\frac{n_s - 1}{n_s}W + \frac{1}{n_s}B)}, \tag{17}$$

jossa

$$V_t = \frac{1}{m(n_s - t)}\sum_{j=1}^{m}\sum_{i=t+1}^{n_s}(\theta_{ij} - \theta_{i-t,j})^2.$$

Tämän estimaatin käyttäminen suoraan n_e laskemisessa osoittautuu hankalaksi sen takia, että autokorrelaatioiden $\hat{\rho}_t$ kohina kasvaa, kun t kasvaa. Tämän takia laskuissa käytetään osasummaa, jolloin n_e estimaatiksi saadaan ([1] s.287)

$$\hat{n}_e = \frac{n_s m}{1 + 2\sum_{t=1}^{T}\hat{\rho}_t}, \tag{18}$$

jossa $T = \min\{t \mid \hat{\rho}_{t+1} + \hat{\rho}_{t+2} < 0\}$. Kuten $\hat{R}$-tilastossa, tarkkaa yksittäistä hyvää arvoa $\hat{n}_e$:lle ei voi määrittää, mutta lähteessä ([1] s.287) suositellaan jatkamaan simulaatioita, kunnes $\hat{n}_e > 5m$, kun m on ketjujen lukumäärä niiden puolittamisen jälkeen.

Näiden lisäksi saatetaan tarkastella myös ehdokkaiden hyväksymisprosenttia. Tämänkin tarkastelu on hyvin tilannekohtaista ja siihen vaikuttaa muun muassa tutkittavan parametrivektorin dimensioiden määrä. Kuitenkin esimerkiksi Metropolis-algoritmin tapauksessa voidaan yksiuloitteisessa tapauksessa pitää hyväksymisprosenttia 44 % hyvänä. Optimaalinen hyväksymisprosentti laskee, kun parametrien määrä kasvaa, ja kun parametrejä on yli 5, tavoiteltava hyväksymisprosentti on noin 23 % ([1] s. 296). Mikäli hyväksymisprosentti on hyvin korkea, on syytä kasvattaa ehdokasjakauman keskihajontaa, ja mikäli liian matala, voidaan keskihajontaa laskea, jotta päästään halutulle tasolle.

4 Esimerkkitoteutukset

4.1 Algoritmin vertailu suoraan otantaan

Esitellään seuraavaksi Metropolis-Hastings-algoritmin toteutus ja vertaillaan sitä suoraan otantaan posteriorijakaumasta lineaarisen regression tapauksessa R-ohjelmointikielellä. Toteutuksen aineistona toimii itse generoitu keinotekoinen data, joka on tehty lineaarisen regression puitteisiin sopivaksi. Otoskoko on $n = 100$ ja selittäviä muuttujia on kaksi vakiotermin lisäksi. Selittävät muuttujat on generoitu jakaumasta $x_{i1} \sim N(2, 4)$ ja $x_{i2} \sim U(0, 6)$ kaikille $i \in \{1, ..., n\}$. Todelliset $\boldsymbol{\beta}$-parametrien arvot ovat kiinnitettyjä, ja ne ovat

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} 1.0 \\ 4.7 \\ -2.2 \end{bmatrix}$$

ja jokaiseen vastemuuttujaan y_i lisätään vielä kohinatermi $\epsilon_i \sim N(0, \sigma^2)$. Tämä varianssi $\sigma^2 = 1$, ja se oletetaan prosessissa tunnettuksi. Eli kaikkineen

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

joka on avattuna

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1, x_{11}, x_{12} \\ 1, x_{21}, x_{22} \\ \vdots \\ 1, x_{n1}, x_{n2} \end{bmatrix} \begin{bmatrix} 1.0 \\ 4.7 \\ -2.2 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}.$$

Asetetaan tasapriori, ja koska varianssi on tunnettu, on todellinen posteriorijakauma esimerkin 1 mukainen $\boldsymbol{\beta} \mid D \sim N(\hat{\boldsymbol{\beta}}, \sigma^2(\mathbf{X}^T\mathbf{X})^{-1})$, ja koska $\sigma^2 = 1$, niin

$$\boldsymbol{\beta} \mid D \sim N(\hat{\boldsymbol{\beta}}, (\mathbf{X}^T\mathbf{X})^{-1}). \tag{19}$$

Tarkoituksena on vertailla Metropolis-Hastings-algoritmin suoriutumista suoraan otantaan kyseisestä posteriorijakaumasta. Algoritmin tapauksessa ehdokasjakaumana on normaalijakauma, koska sen määrittelyjoukkoon kuuluu kaikki reaaliluvut, eikä ole mitään perusteltua syytä suosia muutakaan jakaumaa. Odotusarvo on nykyinen tila ja varianssit ovat $\text{var}(\beta_0) = 0.01$, $\text{var}(\beta_1) = 0.0064$, $\text{var}(\beta_2) = 0.0025$. Näihin varianssien valintaan päädyttiin tarkasteltaessa hyväksymisprosentteja ja $\hat{R}$-arvoja. Kovarianssit puolestaan oletetaan 0:ksi, eli vektorimuodossa

$$\boldsymbol{\beta}^* \sim N(\boldsymbol{\beta}^{t-1}, \boldsymbol{\Sigma}), \tag{20}$$

jossa

$$\boldsymbol{\Sigma} = \begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.0064 & 0 \\ 0 & 0 & 0.0025 \end{bmatrix}.$$

Tämä ehdokasjakauma on symmetrinen, joten kyseessä oleva hyväksymistodennäköisyys on kaavan 9 mukainen

$$\mathcal{A}(\boldsymbol{\beta}^{t-1}, \boldsymbol{\beta}^*) = \min\{1, \frac{p(\boldsymbol{\beta}^* \mid D)}{p(\boldsymbol{\beta}^{t-1} \mid D)}\} = \min\{1, \frac{p(D \mid \boldsymbol{\beta}^*)p(\boldsymbol{\beta}^*)}{p(D \mid \boldsymbol{\beta}^{t-1})p(\boldsymbol{\beta}^{t-1})}\},$$

ja priorijakauma on vakio, eli $p(\boldsymbol{\beta}^*) = p(\boldsymbol{\beta}^{t-1})$, jolloin

$$\mathcal{A}(\boldsymbol{\beta}^{t-1}, \boldsymbol{\beta}^*) = \min\{1, \frac{p(D \mid \boldsymbol{\beta}^*)}{p(D \mid \boldsymbol{\beta}^{t-1})}\}. \tag{21}$$

Alkuarvot tulee $p_0(\boldsymbol{\beta}) \sim N(0, \mathbf{I}_p)$-jakaumasta.

Algoritmi ajettiin 4 kertaa, jossa kussakin ketjussa oli 10000 havaintoa ennen sisäänajojakson poistamista. Hyväksymisprosentit neljässä ketjussa oli desimaalin tarkkuudella [31.8 %, 31.3 %, 30.1 %, 31.1 %] ja parametrien $\hat{R}$:n arvoiksi tuli kolmen desimaalin tarkkuudella [1.001, 1.002, 1.003]. Näiden nojalla voidaan olettaa, että ketjut ovat supenneet tarpeeksi hyvin. *Posteriorivälit* ovat nyt 95 %-posteriorivälejä, jotka ovat laskettu niin, että tarkastellaan [2.5 %, 97.5 %] kvantiiliväliä. Toisin sanoen alaraja on simulaatioiden pienin luku kun pienimmät 2.5 % on poistettu ja yläraja on simulaatioiden suurin luku kun suurimmat 2.5 % on poistettu. Huomaa Metropolis-algoritmin tapauksessa, että simulaatioita tehtiin 4 kappaletta $\hat{R}$ laskemiseksi, mutta nyt tarkastellaan nimenomaan ensimmäistä ketjua. Alla taulukko posteriorivälestä, kun sekä algoritmista, että suorasta otannasta on 5000 havaintoa.

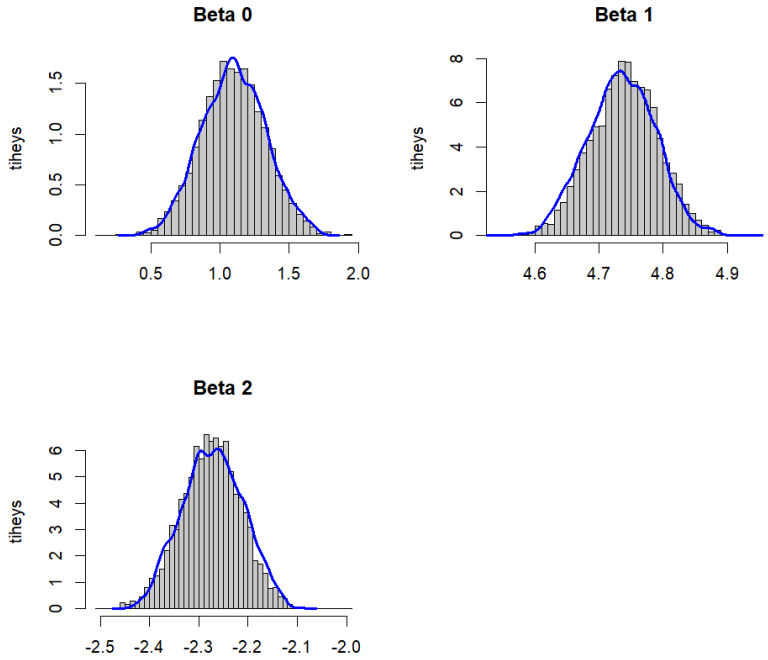
Suora otanta	**Keskiarvo**	**Alaraja (2.5%)**	**Ylaraja (97.5%)**
β_0	1.09	0.64	1.55
β_1	4.74	4.64	4.84
β_2	-2.28	-2.40	-2.16
Metropolis-Hastings	**Keskiarvo**	**Alaraja (2.5 %)**	**Ylaraja (97.5 %)**
β_0	1.10	0.66	1.57
β_1	4.73	4.63	4.83
β_2	-2.27	-2.39	-2.15

Molemmissa tapauksissa etenkin β_1 ja β_2 arvioiminen on mennyt melko onnistuneesti verrattuna todellisiin arvoihin $\beta_1 = 4.7, \beta_2 = -2.2$. Vakiotermin β_0:n tapauksessa keskiarvot ovat melko lähellä todellista arvoa $\beta_0 = 1.1$, mutta posteriorivälit ovat melko laajat. Ennen kaikkea kuitenkin mielenkiinto oli Metropolis-Hastings-algoritmin suoriutumiskyvyssä verrattuna suoraan otantaan, ja näiden valossa se suoriutui varsin hyvin. Kuvassa 1 voi lisäksi nähdä hieman tarkemmin kuvaajat molemmista otannoista, kun ne ovat normalisoitu tiheysfunktioiksi.

4.2 Algoritmin toteutus Housing-datalla

Esitellään seuraavaksi Metropolis-Hastings-algoritmin toteutus Kaggle-verkkosivulta löytyvältä ainestolla "California Housing prices"[13]. Aineisto on lähteen mukaan vuoden 1990 Californian väestönlaskennasta. Aineistossa on kymmenen muuttujaa, mutta joukossa on muuttujia liittyen sijaintiin, jotka eivät ole nyt mielenkiinnon kohteena. Lisäksi muuttuja total_bedrooms poistettiin, koska sillä on vahva korrelaatio muuttujan total_rooms kanssa. Alla käytettävät muuttujat:

Muuttuja	**Selitys**	**Arvojoukko**
median_house_value	Korttelin sisällä olevien asuntojen mediaanihinta (USD)	Pos. reaaliluku
housing_median_age	Korttelin sisällä olevien asuntojen keski-ikä	Pos. kokonaisluku
total_rooms	Huoneiden kokonaismäärä korttelin sisällä	Pos. kokonaisluku
population	Korttelin sisällä asuvien kokonaismäärä	Pos. kokonaisluku
households	Kotitalouksien lukumäärä korttelin sisällä	Pos. kokonaisluku
median_income	Kotitalouksien mediaanitulot kymmennissä tuhansissa (USD) korttelin sisällä	Pos. reaaliluku



Kuva 1: Parametrien jakautumat luvun 4.1 simulaatioiden aikana. Harmaa histogrammi on suora otanta, sininen viiva Metropolis-Hastings.

Vastemuuttuja on "median_house_value", ja loput 5 muuttujaa toimii selittävinä muuttujina. Parametreja on $p = 6$, kun otetaan vakioparametri huomioon. Havaintoja datassa on $n = 20640$.

Huomaa, että varianssiparametri on tuntematon, joten sitä täytyy myös estimoida. Käytetään $\boldsymbol{\beta}$:lle jälleen tasaprioria ja varianssille puolestaan aiemmin mainittua prioria σ^{-2}. Alkuarvot tulee $\boldsymbol{\beta}$-vektorille jakaumasta

$$\boldsymbol{\beta} \sim N(0, \mathbf{I}_p), \tag{22}$$

ja varianssille jakaumasta

$$\sigma^2 \sim N(100, 100). \tag{23}$$

Tässä esimerkissä molempien parametrien ehdokasjakaumat ovat normaalijakaumia, jossa odotusarvo on nykyinen tila. $\boldsymbol{\beta}$:n ehdokasjakauma on

$$\boldsymbol{\beta}^* \sim N(\boldsymbol{\beta}^{t-1}, \boldsymbol{\Sigma}), \tag{24}$$

jossa kovarianssimatriisi on diagonaalimatriisi

$$\Sigma = \text{diag}(49, 2, 1, 1, 1, 2).$$

σ^2:n ehdokasjakauma on puolestaan

$$\sigma^2_* \sim N(\sigma^2_{t-1}, 10000). \tag{25}$$

Näihin kovarianssimatriisin valintoihin päädyttiin diagnostisissa tarkasteluissa. Hyväksymistodennäköisyys on siis $\boldsymbol{\beta}$:n tapauksessa sama kaavan 21 mukainen kuin

edellisessä esimerkissä, ainoana erona ehdokasjakauman kovarianssimatriisi. σ^2:n hyväksymistodennäköisyys on puolestaan

$$\mathcal{A}(\sigma^2_{t-1}, \sigma^2_*) = \min\{1, \frac{p(D \mid \sigma^2_*)p(\sigma^2_*)}{p(D \mid \sigma^2_{t-1})p(\sigma^2_{t-1})}\} = \min\{1, \frac{p(D \mid \sigma^2_*)\sigma^2_{t-1}}{p(D \mid \sigma^2_{t-1})\sigma^2_*}\}. \quad (26)$$

Nyt kun kyseessä on sekä $\boldsymbol{\beta}$:n, että σ^2:n otanta, suoritetaan algoritmikin kahdessa vaiheessa, kuten alla:

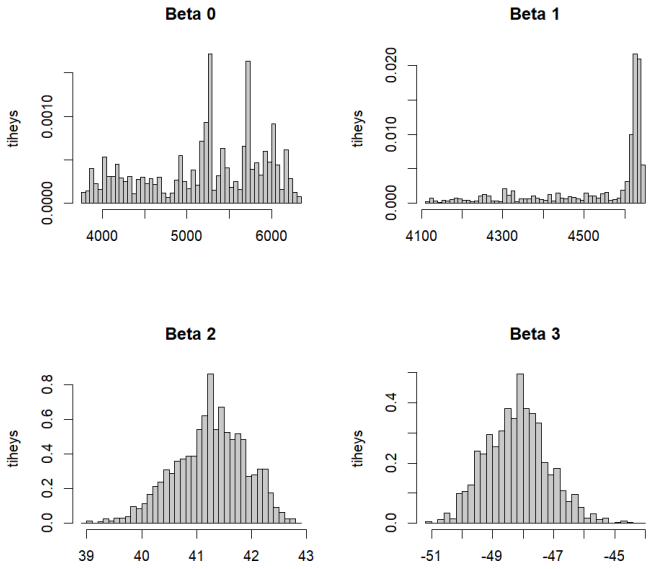
Askel	
1	Sama kuin aiemmin, mutta alkuarvo tulee määrittää sekä $\boldsymbol{\beta}$:lle, että σ^2:lle.
2.1	Suoritetaan otanta ehdokasjakaumasta $\boldsymbol{\beta}^* \sim q(\boldsymbol{\beta}^* \mid \boldsymbol{\beta}^{t-1}, \sigma^2_{t-1})$.
2.2	Lasketaan $\mathcal{A}(\boldsymbol{\beta}^{t-1}, \boldsymbol{\beta}^* \mid \sigma^2_{t-1})$.
2.3	Samalla tavalla kuin aiemmin
2.4	Samalla tavalla kuin aiemmin, paitsi mennään uuteen askeleeseen 3.1
3.1	Suoritetaan otanta ehdokasjakaumasta $\sigma^2_* \sim q(\sigma^2_* \mid \sigma^2_{t-1}, \boldsymbol{\beta})$
3.2	Lasketaan $\mathcal{A}(\sigma^2_{t-1}, \sigma^2_* \mid \boldsymbol{\beta}^t)$
3.3	Samalla tavalla kuin askel 2.3
3.4	Samalla tavalla, kuin askel 2.4. Palataan askeleeseen 2.1.

Huomaa, että tässä parametrivektoria $\boldsymbol{\theta} = (\boldsymbol{\beta}, \sigma^2)$ päivitetään osissa kuin Gibbsin otannassa, mutta hyväksymisprosessi on kuin Metropolis-Hastings algoritmissa ja tämän voi nähdä näiden kahden yhdistelmänä.

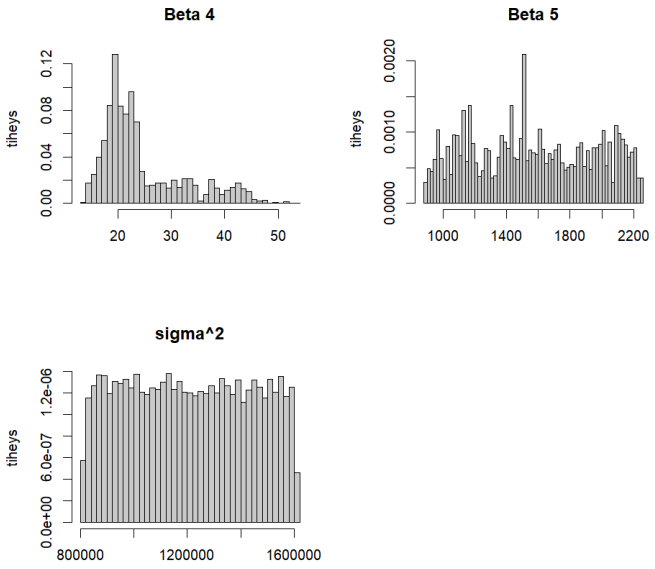
Algoritmi ajettiin 4 kertaa, joissa kussakin simulaatiossa on aloitusotanta mukaan lukien 40000 havaintoa. Sisäänajojakson poiston jälkeen havaintoja jää 20000. Hyväksymisprosentit olivat $\boldsymbol{\beta}$-vektorin tapauksessa [12.6 %, 12.7 %, 12.7 %, 12.7 %], jotka ovat hieman tavoitetasoa matalampia, mutta toimivia. $\hat{R}$-arvot olivat puolestaan [1.076, 1.021, 1.022, 1.004, 1.025, 1.023].

Varianssiparametrin σ^2 tapauksessa hyväksymisprosentit olivat [50.0 %, 49.7 %, 50.2 %, 50.5 %], jotka ovat hieman tavoitetasoa korkeampia, mutta tarpeeksi hyviä. $\hat{R}$-arvo oli puolestaan kolmen desimaalin tarkuudella 1.001, joka on todella hyvä. Alla taulukossa on posteriorivälit neljän merkitsevän numeron tarkkuudella ensimmäiselle ketjulle.

	Keskiarvo	**Alaraja (2.5 %)**	**Ylaraja (97.5 %)**
β_0	5212	3874	6197
β_1 housing_median_age	4535	4174	4642
β_2 total_rooms	41.25	39.88	42.37
β_3 population	-48.21	-50.10	-46.13
β_4 households	24.46	15.30	43.56
β_5 median_income	1571	940.0	2204
σ^2	1207000	830800	1588000



Kuva 2: Parametrien $\beta_0, \beta_1, \beta_2, \beta_3$ jakautumat luvun 4.2 toteutuksessa



Kuva 3: Parametrien $\beta_4, \beta_5, \sigma^2$ jakautumat luvun 4.2 toteutuksessa

5 Yhteenveto

Tutkielmassa tutustuttiin bayesilaiseen lineaariseen regressioon ja esiteltiin sen yhteydessä MCMC-menetelmiä. Luvussa 2 tarkasteltiin bayesilaisen ja frekventistisen tilastotieteen eroja sekä esiteltiin lineaarista regressioanalyysia molemmista näkökulmista.

Luvussa 3 esiteltiin MCMC-menetelmiä, joiden avulla voidaan suorittaa otantaa bayesilaisesta posteriorijakaumasta. Tämän tueksi esiteltiin Markov-ketjujen teoriaa sekä tarkasteltiin matemaattisia perusteluita Metropolis-Hastings-algoritmin suppenemiselle kohti posteriorijakaumaa.

Luvussa 4 verrattiin Metropolis-Hastings-algoritmin suoriutumista suoraan otantaan keinotekoisella datalla. Suora otanta toimi vertailukohtana, sillä se perustuu samaan priorijakaumaan ja uskottavuusfunktioon kuin algoritmi, ja se tuottaa odotusarvoisesti parhaita otoksia posteriorijakaumasta näillä priorin ja uskottavuusfunktion oletuksilla. Metropolis-Hastings-algoritmi suoriutui vertailussa hyvin, mikä viittaa siihen, että algoritmi kykenee tehokkaaseen posterioriotantaan. Lisäksi luvussa 4 tutkittiin algoritmin suoriutumista oikealla datalla.

Viitteet

[1] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, D. B. Rubin: *Bayesian data analysis* (3. painos, päivitetty 2021)

[2] A. Gelman: *The Development of Bayesian Statistics* (Artikkeli, 2022) Saatavilla: `https://link.springer.com/article/10.1007/s41745-022-00307-y`

[3] I. Fornacon-Wood, H. Mistry, C. Johnson-Hart, C. Faivre-Finn, J. P.B. O'Connor, G. J. Price: *Understanding the Differences Between Bayesian and Frequentist Statistics* (Artikkeli, 2021) Saatavilla: `https://www.redjournal.org/article/S0360-3016(21)03256-9/fulltext`.

[4] H. Nyberg: *Lineaariset ja yleistetyt lineaariset mallit* (Luentomoniste, Turun Yliopisto, 2024)

[5] J. S. Speagle: *A Conceptual Introduction to Markov Chain Monte Carlo Methods* (Saatavilla `https://arxiv.org/abs/1909.12313)`

[6] R. Douc, E. Moulines, P. Priouret, P. Soulier: *Markov Chains*, Springer Series in Operations Research and Financial Engineering

[7] C. Robert, G. Casella: *Monte Carlo statistical methods* (1. painos, 1999)

[8] S.P. Meyn and R.L Tweedie: *Markov Chains and stochastic stability* (2005)

[9] Bilal Bilal: *Markov Chain Monte Carlo*, Uppsala University (2023)

[10] L. Tierney: *Markov Chains for exploring posterior distributions*, University of Minnesota (1994)

[11] C. Andrieu, N. De Freitas, A. Doucet, M. I. Jordan: *An Introduction to MCMC for Machine Learning*

[12] `https://sanasto.tilastoseura.fi/`

[13] `https://www.kaggle.com/datasets/camnugent/california-housing-prices`

[14] `https://cran.r-project.org/web/packages/posterior/vignettes/posterior.html`

Liitteet

Alla R-koodi luvun 4.1 toteutukseen. Lisätietoa posterior-kirjastosta löytyy lähteestä [14].

```
# Kirjastot
library(MASS)     # Multinormaalijakaumaa varten
library(posterior) # R-hatun laskemista varten

# Alustetaan satunnaislukugeneraattori jollain arvolla
set.seed(127)


######################
# 1. Suoritetaan datan generointi:
# Generoidaan muuttujien x1 ja x2 arvot ja tallenetaan vektoreiksi.
# Muodostetaan matriisi X, jossa:
# ensimmäinen sarake koostuu ykkösistä,
# toinen sarake on x1 vektori,
# kolmas sarake x2 vektori
# Asetetaan todelliset beta-parametrien arvot vektoriin beta_true
# Generoidaan "kohinatermit" N(0,1) jakaumasta,
# Tallenetaan ne vektoriin error
# Asetetaan y = X*true_beta + kohinatermi
######################


n <- 100  # "Otoksen" koko
# Generoidaan x1:t N(2,4)-jakaumasta
x1 <- rnorm(n, mean = 2, sd = 2)   # huomaa sd = keskihajonta
# Generoidaan x2:t U(0,6)-jakaumasta
x2 <- runif(n, min = 0, max = 6)

# Muodostetaan selittävien muuttujien matriisi
# ensimmäinen sarake koostuu vain ykkösistä vakiotermin takia
X <- cbind(1, x1, x2)  # Ensimmäinen sarake pelkästään ykkösiä
beta_true <- c(1.0, 4.7, -2.2)  # Todelliset parametrien arvot

# Generoidaan vastemuuttuja. y = X*beta + virhetermi.
# N(0,1)-jakaumasta
error <- rnorm(n, mean = 0, sd = 1)   # Generoidaan kohina
y <- X %*% beta_true + error


############################
# 2. Tehdään funktio joka suorittaa Metropolis-Hastings-algoritmia.
# Käytän logaritmista asteikkoa, koska törmäsin ongelmiin numeerisen
```

```
 # stabiliteetin kanssa.
# Hyväksymistodennäköisyyttä laskiessa huomaa että log(a/b) = log(a)-log(b),
 # joten laskettu erotuksena.
# Hyväksymisehdon laskemistesta:
# acceptance ratio tarkoittaa suhdelukua r laskiessa hyväksymis-
# todennäköisyyttä min{1, r} (katso kaava (9) tutkielmasta).
# huomaa log(1) = 0, eli jos log_acceptance_ratio > 0, niin hyväksytään aina
 # muuten lasketaan exp(log_acceptance_ratio) = r
# Varianssi tunnettu eli 1,
# Ehdokasjakauma (proposal distribution) normaalijakautunut, jossa odotusarvo on
 # nykyinen tila, keskihajonnoilla voi pelata kun kutsuu funktiota.
# Lisäksi tehdään accepts vektori, joka saa arvon 1 mikäli ehdokas
 # hyväksytään, ja arvon 0 mikäli ei. Tämän avulla voidaan laskea
 # ehdokkaiden hyväksymisprosentti.
############################
metropolis_hastings <- function(X, y, sigma2 = 1, num_iterations,
                                proposal_sd) {
  p <- ncol(X)      # Kiinnitetään p:n arvo
  samples <- matrix(0, nrow = num_iterations, ncol = p)

  # Aloituspiste satunnaisesti N(0,1)-jakaumasta
  # Vastaa taulukon algoritmin askelta 1
  beta_current <- rnorm(p, mean = 0, sd = 1)
  # Alustetaan log-uskottavuus tällä betalla
  current_loglik <- -0.5 * sum((y - X %*% beta_current)^2) / sigma2
  # Alustetaan accepts-vektori nollilla
  accepts <- numeric(num_iterations)
  # Tehdään for-loop, joka suorittaa algoritmin askeleita 2.1-2.4:
  for (i in 1:num_iterations) {
    # Uusi betan ehdotus
    # Vastaa askelta 2.1. Katso myös kaava (20)
    beta_proposal <- beta_current + rnorm(p, mean = 0, sd = proposal_sd)

    # Lasketaan hyväksymistodennäköisyys (tutkielma kaava 9 ja 21 ja
     # algoritmin askel 2.2) log-asteikoilla.
    # Tässä betan suhteen vakiot ovat supistettu pois. Lisäksi vakiopriori
     # supistuu pois.
    proposal_loglik <- -0.5 * sum((y - X %*% beta_proposal)^2) / sigma2
    # Tässä siis huomaa log(a)-log(b) = log(a/b)
    log_accept_ratio <- proposal_loglik - current_loglik

    # muunnetaan pois log-asteikolta
    acceptance_prob <- ifelse(log_accept_ratio > 0, 1, exp(log_accept_ratio))

    # Askeleet 2.3 ja 2.4
    u <- runif(1)                   # askel 2.3
```

```
    if (u < acceptance_prob) {      # askel 2.4
      beta_current <- beta_proposal
      current_loglik <- proposal_loglik
      accepts[i] <- 1 # asetetaan accepts vektoriin 1 hyväksymisen merkiksi
    }

    # Lisätään havaintoihin ja mennään takaisin askeleeseen 2.1
    samples[i, ] <- beta_current
  }

  # Tulostetaan ehdokkaiden hyväksymisprosentti
  cat("Hyväksymisprosentti:", mean(accepts), "\n")

  # Iteraatiot tehty: poistetaan ensimmäinen puolisko (sisäänajojakso)
  return(samples[(num_iterations/2):num_iterations, ])
}


##########################################
# 3. MH-algoritmin suppenemisen tarkastelu
# Tehdään neljä ketjua ja lasketaan R-hattu
##########################################
chains <- list() # Alustetaan ketjut listaksi
n_chains <- 4  # Ketju lukumäärä

# Ajetaan MH-algoritmi kaikille ketjuille
# iteraatioita 9999 aloitusotannan lisäksi, jolloin yhteensä 10000
 # sisäänajon poiston jälkeen siis 5000
for (i in 1:n_chains) {
  chains[[i]] <- metropolis_hastings(X, y, num_iterations = 9999,
                                     proposal_sd = c(0.1, 0.08, 0.05))
}

# Muutetaan lista 3D-arrayksi, dimensiot: [iteraatio, parametri, ketju]
# Alustetaan tyhjä array
chains_array <- array(NA, dim = c(5000, 3,
                                  n_chains))
# for-looppi tämän täydentämiseksi
for (i in 1:n_chains) {
  chains_array[,,i] <- chains[[i]]
}
# Alustetaan vektori R-hattu arvojen tallennusta varten
rhat_values <- numeric(3)

# For-loop joka käy kaikki kolme parametria läpi ja laskee niille R-hatun
  # (1 = beta0, 2 = beta1, 3 = beta2)
```

```
for (i in 1:3) {
  # Eristetään yksittäinen parametri 5000 x 1 x 4 matriisiksi
  param_matrix <- chains_array[, i, ]
  # Muutetaan matriisi muotoon jota posterior-kirjaston rhat-funktio ymmärtää
  param_draws <- as_draws_array(param_matrix)
  # Lasketaan R-hattu kyseisellä funktiolla
  rhat_values[i] <- rhat(param_draws)
}

# R-hatut
rhat_values

#################################
# 4. Tehdään suoraa otanta posteriorista 5000 kertaa
#################################

# Tiedetään, että posteriori on N(beta_hat, (X^T * X)^(-1)) (tutkielma (19))
# (katso tutkielman esimerkki 1)
# Lasketaan beta_hat, eli betan SU-estimaatti
XtX_inv <- solve(t(X) %*% X)    # Kiinnitetään (X^T * X)^(-1)
beta_hat <- XtX_inv %*% t(X) %*% y  # betan SU-estimaatti

# Generoidaan nyt havaintoja posteriorista
direct_samples <- mvrnorm(n = 5000, mu = beta_hat, Sigma = XtX_inv)




#################################
# 5. Vertaillaan MH-algoritmin antamia tuloksia suoraan otantaan
# Lasketaan molemmille tapauksille kaikki keskiarvot ja posteriorivälit
# 95%-posterioriväli lasketaan poistamalla pienimmät ja suurimmat 2,5%
# havainnoista
#################################

# Lasketaan keskiarvot ja 95% posterioriväli suoraan otannasta
direct_means <- colMeans(direct_samples)
# 2.5%-kvantiili
direct_025 <- apply(direct_samples, 2, function(x) quantile(x, 0.025))
# 97.5%-kvantiili
direct_975 <- apply(direct_samples, 2, function(x) quantile(x, 0.975))

# Lasketaan keskiarvot ja 95% posterioriväli Metropolis-Hastingsin otannasta
# Huomaa, että Markov-ketjuja generoitiin useampi. Käytetään ensimmäistä.
mh_samples <- chains_array[,,1] # ensimmäinen ketju
mh_means <- colMeans(mh_samples)
```

```
mh_025 <- apply(mh_samples, 2, function(x) quantile(x, 0.025)) # 2.5%-kvantiili
mh_975 <- apply(mh_samples, 2, function(x) quantile(x, 0.975)) # 97.5%-kvantiili

# Tulostetaan keskiarvot ja 95%-posteriorivälit
print("Suoran otannan posterioriväli")
print(data.frame(Keskiarvo = direct_means, Alaraja = direct_025,
                 ylaraja = direct_975))

print("MH posterioriväli")
print(data.frame(Keskiarvo = mh_means, Alaraja = mh_025, Ylaraja = mh_975))



########################################
#6. Plotataan havainnot
########################################


# Tehdään 2x2-ruudukko
par(mfrow = c(2, 2))

# for looppi joka plottaa kaikkien parametrien havainnot suoralla otannalla
# (histogrammi) ja MH-algoritmilla (viiva)
for (param_index in 1:3) {
  # Suoran otannan plottaus
  hist(direct_samples[, param_index],
       main = paste("Beta", param_index-1),
       xlab = paste(""),
       ylab = paste("tiheys"),
       col = rgb(0.3, 0.3, 0.3, 0.3), # Joku sopivalta vaikuttava värikoodi
       probability = TRUE,            # Tämä "normalisoi" tiheysfunktioiksi
       breaks = 50)

  # MH-algoritmin plottaus
  lines(density(mh_samples[, param_index]), col = "blue", lwd = 2)
}
```

Alla R-koodi luvun 4.2 toteutukseen

```
    # Koodissa vähemmän selostusta, koska pitkälti sama logiikka kuin edellisessä
 # esimerkissä

library(MASS)
library(posterior)

# satunnaislukugeneraattorin alustus
```

```
set.seed(127)

# Datan alustus. Tiedosto housing jo importattu Rstudioon.
data <- housing
nrow(data)
# 20640 riviä
# tarkistetaan onko NA-arvoja
anyNA(data)
# poistetaan rivit, joissa NA-arvoja
data <- na.omit(data)
nrow(data)
# NA:ta sisältävien rivien poiston jälkeen 20433 riviä
# tarkistetaan poistettiinko turhia rivejä, eli sisältyykö NA:t muuttujiin,
# jota ei muutenkaan käytetä.
data <- housing
X <- data[, c("housing_median_age", "total_rooms",
              "population", "households", "median_income")]
X <- as.matrix(cbind(1, X)) # lisätään ensimmäiseksi sarakkeeksi ykkösiä
X <- na.omit(X)
nrow(X)
# tässä 20640 riviä, eli NA:ta sisältävät rivit ei sisälly käytettäviin
 # muuttujiin. Voidaan siis jatkaa poistamatta mitään.
data <- housing
# määritetään vastemuuttuja
y <- data$median_house_value
# määritetään selittävät muuttujat
X <- data[, c("housing_median_age", "total_rooms",
              "population", "households", "median_income")]
# Lisätään ensimmäiseksi sarakkeeksi ykkösiä vakiotermin takia
X <- as.matrix(cbind(1, X))

################################
# 1. Kaksivaiheinen MH-funktio:
# Ensiksi otanta beta-vektorille, sen jälkeen sigma^2:lle
# proposal_cov_beta on betan ehdokkaan kovarianssimatriisi
# proposal_sd_sigma2 on sigma^2 parametrin ehdokkaan keskihajonta
# laskenta taas log-asteikolla numeerisen stabiliteetin takia
################################
metropolis_hastings2 <- function(X, y, num_iterations, proposal_cov_beta,
                                 proposal_sd_sigma2) {
  p <- ncol(X)

  beta_samples <- matrix(0, nrow = num_iterations, ncol = p)
  sigma2_samples <- numeric(num_iterations)

  # Alkuarvo. Vastaa askelta 1, katso myös tutkielman kaavat (22) ja (23)
```

```
beta_current <- rnorm(p, 0, 1)
sigma2_current <- rnorm(1, 100, 10)

loglik_current <- -0.5 * sum((y - X %*% beta_current)^2) / sigma2_current

# Alustetaan hyväksymisprosenttivektorit
beta_accepts <- numeric(num_iterations)
sigma2_accepts <- numeric(num_iterations)

# for loop joka iteroi uuden algoritmin askeleita 2-3
for (i in 1:num_iterations) {
  # Betan ehdotus (askel 2.1)
  beta_proposal <- mvrnorm(1, mu = beta_current, Sigma = proposal_cov_beta)

  # askel 2.2
  loglik_proposal <- -0.5 * sum((y - X %*% beta_proposal)^2) / sigma2_current
  log_accept_ratio <- loglik_proposal - loglik_current
  accept_prob <- ifelse(log_accept_ratio > 0, 1, exp(log_accept_ratio))

  u <- runif(1) # askel 2.3
  if (u < accept_prob) {    # askel 2.4
    beta_current <- beta_proposal
    loglik_current <- loglik_proposal
    beta_accepts[i] <- 1
  }

  # Sigma^2 ehdotus (askel 3.1)
  sigma2_proposal <- sigma2_current + rnorm(1, 0, proposal_sd_sigma2)

  # askel 3.2
  if (sigma2_proposal > 0) {
    loglik_proposal <- -0.5 * sum((y - X %*% beta_current)^2)/
      sigma2_proposal
    log_accept_ratio <- (loglik_proposal + log(sigma2_current)) -
      (loglik_current + log(sigma2_proposal))

    accept_prob <- ifelse(log_accept_ratio > 0, 1, exp(log_accept_ratio))

    u <- runif(1) # askel 3.3
    if (u < accept_prob) {    # askel 3.4
      sigma2_current <- sigma2_proposal
      loglik_current <- loglik_proposal
      sigma2_accepts[i] <- 1  # Merkitään ylös
    }
  }
```

```
    beta_samples[i, ] <- beta_current
    sigma2_samples[i] <- sigma2_current
  }

  # muodostetaan listat betan arvoista ja sigma^2 arvoista, poistetaan sisäänajo
  # lisäksi tallenetaan listaa hyväksynnät
  list(
    beta = beta_samples[(num_iterations / 2):num_iterations, ],
    sigma2 = sigma2_samples[(num_iterations / 2):num_iterations],
    beta_accepts = beta_accepts,
    sigma2_accepts = sigma2_accepts
  )
}

# Ajetaan MH neljä kertaa
chains <- list()
for (i in 1:4) {
  chains[[i]] <- metropolis_hastings2(X, y, num_iterations = 39999,
                                      proposal_cov_beta <- diag(c(49, 2, 1,
                                                                  1, 1, 2)),
                                      proposal_sd_sigma2 = 100)
}

# Muutetaan beta-lista 3D-arrayksi, dimensiot: [iteraatio, parametri, ketju]
# Alustetaan tyhjä array
beta_array <- array(0, dim = c(20000, 6,
                               4))

# Looppi joka täydentää beta-arvot
for (i in 1:4) {
  beta_array[, , i] <- chains[[i]]$beta
}
# muutetaan sigma2-lista 3D-arrayksi, dimensiot: [iteraatio, 1, ketju]
# alustetaan tyhjä array
sigma2_array <- array(0, dim = c(20000, 1, 4))

# Looppi jolla täydennetään sigma^2 arvot
# pitää erikseen muuttaa matriisiksi
for (i in 1:4) {
  sigma2_array[, , i] <- matrix(chains[[i]]$sigma2)
}

# Lasketaan R-hattu beta-parametreille
rhat_beta <- numeric(6)
for (i in 1:6) {
  param_matrix <- beta_array[, i, ]
```

```
  param_draws <- as_draws_array(param_matrix)
  rhat_beta[i] <- rhat(param_draws)
}

# R-hattu sigma^2:lle
sigma2_draws <- as_draws_array(sigma2_array)
rhat_sigma2 <- rhat(sigma2_draws)

# Hyväksymisprosentit
beta_accepts <- unlist(lapply(chains, function(x) mean(x$beta_accepts)))
sigma2_accepts <- unlist(lapply(chains, function(x) mean(x$sigma2_accepts)))

# Tulokset:
beta_accepts
sigma2_accepts

rhat_beta
rhat_sigma2


# Posteriorivälien laskenta
# beta-parametrille:
samples_beta <- beta_array[,,1] # Käytetään ensimmäistä ketjua
means_beta <- colMeans(samples_beta) # Keskiarvo
# 2.5%-kvantiili:
beta_025 <- apply(samples_beta, 2, function(x) quantile(x, 0.025))
# 97.5%-kvantiili;
beta_975 <- apply(samples_beta, 2, function(x) quantile(x, 0.975))

# sigma^2:lle:
samples_sigma2 <- sigma2_array[,,1]  # Käytetään ensimmäistä ketjua
means_sigma2 <- mean(samples_sigma2)  # Keskiarvo
sigma2_025 <- quantile(samples_sigma2, 0.025)  # 2.5%-kvantiili
sigma2_975 <- quantile(samples_sigma2, 0.975)  # 97.5%-kvantiili

# tulostetaan vastaukset
print("beta posterioriväli:")
print(data.frame(Keskiarvo = means_beta, Alaraja = beta_025,
                 Ylaraja = beta_975))
print("sigma^2 posterioriväli:")
print(data.frame(Keskiarvo = means_sigma2, Alaraja = sigma2_025,
                 Ylaraja = sigma2_975))


# Plottaus
par(mfrow = c(2, 2))
```

```
# beta
for (param_index in 1:6) {
  hist(samples_beta[, param_index],
       main = paste("Beta", param_index-1),
       xlab = paste(""),
       ylab = paste("tiheys"),
       col = rgb(0.3, 0.3, 0.3, 0.3),
       probability = TRUE,
       breaks = 50)

}

#sigma^2
hist(samples_sigma2,
     main = paste("sigma^2"),
     xlab = paste(""),
     ylab = paste("tiheys"),
     col = rgb(0.3, 0.3, 0.3, 0.3),
     probability = TRUE,
     breaks = 50)
```