Mukautuvan koeasetelman soveltaminen kategorisen havaitsemisen tutkimiseen neuropsykologiassa

Minna Lehtomäki

Tilastotieteen pro gradu -tutkielma

Jyväskylän yliopisto
Matematiikan ja tilastotieteen laitos
17. syyskuuta 2014

Esipuhe

Aloitin pro gradu -tutkielmani tekemisen jo syksyllä 2012 ja sainkin tutkielman melko nopeassa tahdissa etenemään: vuodenvaihteessa ohjelma oli pieniä muokkauksia lukuunottamatta tehty ja itse kirjoitustyökin jo aluillaan. Keväällä 2013 synnytin esikoiseni, jolloin tutkielman tekemiseen tuli äitiyslomani mittainen tauko. Jatkoin tutkielmani parissa uudelleen keväällä 2014, jolloin ohjelman viimeistely, koehenkilöiden värvääminen sekä mittaukset suoritettiin melko nopealla aikataululla. Kesällä 2014 suurin osa työstä olikin jo takanapäin ja enää oli jäljellä pelkkää kirjoitustyötä. Nyt on sekin sitten viimein jo ohi ja tutkielma valmis.

Kokonaisuudessaan tähän projektiin kuului jonkin verran turhautumista, hermojen menettämistä sekä ajoittaista epätoivoa. Suttupaperia kului enemmän kuin tarpeeksi kaavoja pyöritellessä ja herkkuja tuli syötyä stressiin useammin kuin kehtaan myöntää. Onneksi kuitenkin tutkielman tekeminen tarjosi myös uusia oivalluksia ja aitoa iloa siitä, että oli ihan itse osannut pitkän pohtimisen jälkeen ratkaista jonkin ongelman. En olisi selvinnyt loppuun asti ilman apua ja erityisen suuri kiitos kuuluukin miehelleni, joka jaksoi kannustaa eteenpäin aina haasteissa, väsymyksen kynnyksellä ja erityisesti oman uskoni jo horjuessa.

Olen kiitollinen myös ohjaajilleni Juha Karvaselle tilastotieteen laitokselta ja Tiina Parviaiselle psykologian laitokselta. Juha oli aina valmis auttamaan ja vastaamaan kysymyksiini (erityisesti niihin tyhmiin), lisäksi hänellä oli ehtymätön varasto todella hyviä ideoita, jotka tulivat tarpeeseen vaikean paikan sattuessa. Tiina puolestaan tarjosi todella mielenkiintoisen aiheen, lisäsi ymmärrystäni neuropsykologiasta sekä oli jokaisessa tapaamisessa innokkaana keskustelemassa toteutuksesta ja heittelemässä ideoita. Suuri kiitos kuuluu myös Jan Wikgrenille ja Suvi Karlalle psykologian laitokselta. Jan toimi tutkielmani loppuvaiheessa Tiinan sijaisena, oli apuna kaikissa teknisissä asioissa ja auttoi minua saamaan kaiken mittauksia varten toimintakuntoon. Suvi puolestaan oli korvaamaton apu EEG-mittauksissa, joita en olisi osannut yksin suorittaa. Lopuksi haluaisin kiittää myös kaikkia niitä nimeltä mainitsemattomia henkilöitä, jotka eivät edes ole itse tajunneet auttaneensa minua tutkielmani kanssa.

Tiivistelmä

Lehtomäki, Minna: Mukautuvan koeasetelman soveltaminen kategorisen havaitsemisen tutkimiseen neuropsykologiassa
Tilastotieteen pro gradu -tutkielma, Jyväskylän yliopisto, 17. syyskuuta 2014
Sivuja 42, liitteitä 7 (40 sivua)

Tässä tutkielmassa tutkitaan kognitiiviseen neuropsykologiaan kuuluvaa ilmiötä, nimeltään äänteiden kategorinen havaitseminen, käyttämällä mukautuvaa koeasetelmaa. Tarkoituksena on ollut kehittää ohjelma, jonka avulla pystyttäisiin mahdollisimman nopeasti ja tehokkaasti (optimaalisesti) määrittämään jokaiselle koehenkilölle yksilöllinen malli, jota havainnot noudattavat.

Äänteiden kategorinen havaitseminen tarkoittaa, että ihminen pyrkii luokittelemaan äänteet kategorioihin niiden akustisten ominaisuuksien perusteella ilman, että hän erottaisi kahden eri äänteen välimuotoja. Esimerkiksi kuullessaan äänteen, joka on puoliksi /ba/ ja puoliksi /pa/, ihminen ei erota kuultua äännettä välimuodoksi vaan tulkitsee sen joko äänteeksi /ba/ tai /pa/.

Tässä tutkielmassa tarkoituksena on mallintaa, miten koehenkilöt kuulevat äänteet /ba/ ja /pa/ sekä näiden välimuodot ja mitata aivosähkökäyrässä (EEG) tapahtuvia muutoksia koe- ja kontrollitilanteessa. Koetilanne suoritetaan toistamalla koehenkilöille useita kertoja 9 erilaista ääntä ja mittaamalla, kumpi äänne oli koehenkilön mielestä kyseessä. Vasteen (koehenkilön vastausten) oletetaan noudattavan logistista regressiomallia. Kontrollitilanteessa puolestaan koehenkilö pelkästään passiivisesti kuuntelee ääniä.

Koetta varten on tehty MATLAB-ohjelma, joka pyrkii valitsemaan kolme niin sanottua ”kiinnostavaa ääntä”, joiden kohdalla koehenkilö on epävarmin, kummasta äänteestä on kyse. Näiden kiinnostavien äänten määrää painotetaan toistamalla yksittäistä kiinnostavaa ääntä kaksinkertainen määrä verrattuna yksittäiseen ”ei-kiinnostavaan ääneen”, jolloin kolmea kiinnostavaa ääntä toistetaan yhteensä saman verran kuin kuutta ei-kiinnostavaa ääntä. Apuna kiinnostavien äänten valitsemisessa on binäärinen etsintäalgoritmi, joka pyrkii puolitushaulla löytämään kaksi kiinnostavaa ääntä. Kun kaksi ääntä on löydetty, algoritmi jatkaa mallin estimoimista suurimman uskottavuuden menetelmällä kaikesta kerätystä aineistosta.

Tätä tutkielmaa varten on suoritettu mittauksia koehenkilöillä, joiden perusteella ohjelman toimintaa ja kehittämiskohtia on analysoitu. Samalla on kerätty EEG-dataa psykologian laitoksen analysoitavaksi. Suoritettujen mittausten perusteella ohjelma näyttäisi toimivan melko hyvin ja on hyvä pohja koetilanteen kehittämiseksi vielä optimaalisemmaksi.

Avainsanoja: binäärinen etsintäalgoritmi, D-optimaalisuus, kategorinen havaitseminen, koesuunnittelu, kognitiivinen neuropsykologia, logistinen regressio, MATLAB, mukautuva koeasetelma

Sisällys

1 Johdanto 1

2 Binäärisen vasteen teoriaa 5

2.1 Logistinen regressiomalli 5

2.1.1 Mallin johtaminen 6

2.1.2 Parametrien estimointi 7

2.2 Optimaaliset mallit ja *D*-optimaalisuus 10

2.2.1 Yleisesti 10

2.2.2 *D*-optimaalinen malli logistiselle regressiolle 13

2.3 Suurimman uskottavuuden estimaattien olemassaoloehto 15

2.3.1 Täysi erillisyys 16

2.3.2 Näennäinen erillisyys 17

2.3.3 Päällekkäisyys 17

2.4 Binäärinen etsintäalgoritmi 18

3 Kategorisen havaitsemisen tutkiminen 22

3.1 Äänteiden kategorinen havaitseminen 22

3.2 Koeasetelma 23

3.2.1 Koe- ja kontrollikäsittelyt 23

3.2.2 Kritiikkiä koeasetelmaa koskien 24

3.2.3 Tavoitteet 24

4 Ohjelma kategorisen havaitsemisen tutkimiseen 26

4.1 Pääohjelma – perceptionTest 26

4.2 Aliohjelmat 29

4.2.1 CalculatePOI 29

4.2.2 DrawFigure 30

4.2.3 Optdesign 30

4.2.4 PlaySounds ja playThese 31

4.2.5 RandomizeSoundsLeft 32

5 Mittaukset ja ohjelman toiminta 33

5.1 Mittaukset 33

5.2 Ohjelman toiminnan analysointi 34

6 Pohdintaa 41

Lähteet 43
Liitteet 45
Liite 1: Ohjelman tuottamat kuvaajat 45
Liite 2: Esimerkit excel-tallennuksista 50
Liite 3: Koehenkilöiltä kerätyt taustatiedot 52
Liite 4: Pääohjelman MATLAB-koodi 53
Liite 5: Aliohjelmien ohjelmakoodit 62
Liite 6: draw-ohjelman MATLAB-koodi 75
Liite 7: cprintf-ohjelman MATLAB-koodi 76

1 Johdanto

Neuropsykologia on psykologian erikoisala, joka pyrkii ymmärtämään ja tutkimaan aivojen toiminnan ja käyttäytymisen välistä yhteyttä. Erityisesti halutaan tarkastella, mitkä aivojen mekanismit vastaavat erilaisista toiminnoista, esimerkiksi tunteiden säätelystä ja oppimisesta sekä miten nämä mekanismit toimivat. Tässä pro gradu -tutkielmassa erityisenä kiinnostuksen kohteena on niin kutsuttu kognitiivinen neuropsykologia, joka tutkii kognitiivisten prosessien ja aivojen toiminnan välistä yhteyttä. Kognitiivisia prosesseja ovat kaikki tiedon käsittelyyn liittyvät toiminnot, esimerkiksi muisti, oppiminen ja havaitseminen. (Beaumont, 2008).

Neuropsykologia jaetaan yleensä kahteen eri suuntaukseen: kliiniseen ja kokeelliseen neuropsykologiaan. Kliinisessä neuropsykologiassa tutkitaan, diagnosoidaan, hoidetaan ja kuntoutetaan ihmisiä, joilla on aivojen toimintaan liittyviä sairauksia tai vammoja. Kokeellisessa neuropsykologiassa puolestaan tutkitaan yleensä terveitä ihmisiä laboratorioissa. Yleensä koehenkilö suorittaa jonkinlaisia tehtäviä, esimerkiksi keskittymiseen, havainnointiin tai kehonhallintaan liittyen samalla, kun hänen aivojensa toimintaa tarkkaillaan jollakin aivojen kuvantamismenetelmällä. Näitä menetelmiä ovat muun muassa aivosähkökäyrä (*electroencephalography*, EEG), aivomagneettikäyrä (*Magnetoencephalography*, MEG) ja magneettikuvaus (*Magnetic Resonance Imaging*, MRI). Kuvantamismenetelmillä saadut tulokset tallennetaan, jotta ne voidaan myöhemmin analysoida. (Beaumont, 2008).

Tässä tutkielmassa paino on nimenomaan kokeellisessa kognitiivisessa neuropsykologissa ja käytössä on tutkimusmenetelmä, jota kutsutaan kokeelliseksi tutkimukseksi. Kokeellisessa tutkimuksessa tutkitaan syy-seuraussuhteita eli miten jokin muuttuja x vaikuttaa johonkin toiseen muuttujaan y, esimerkiksi melun vaikutusta keskittymiseen. Syy-seuraussuhteen tutkiminen tapahtuu muuttamalla muuttujaa x (melun määrä) ja mittaamalla muutoksen vaikutusta muuttujaan y (keskittymiskyky). Edellä olevassa esimerkissä melun ja keskittymiskyvyn välistä syy-seuraussuhdetta voidaan tutkia siten, että jaetaan koehenkilöt kahteen ryhmään, joista ensimmäisen ryhmän jäsenet suorittavat keskittymistä mittaavia tehtäviä hiljaisessa tilassa ja toisen ryhmän jäsenet suorittavat tehtäviä meluisassa tilassa. Ensimmäinen ryhmä on tällöin niin sanottu kontrolliryhmä, jonka kohdalla syy-muuttuja x saa tietyn arvon (ei melua). Toinen ryhmä puolestaan on koeryhmä, jolla syy-muuttuja (melun määrä) vaihtelee verrattuna kontrolliryhmään. Tämä ilmaistaan yleensä sanomalla, että koeryhmä on saanut tutkittavan käsittelyn ja kontrolliryhmä puolestaan ei ole saanut käsittelyä.

Joissakin tilanteissa, esimerkiksi juuri neuropsykologiassa, tällainen koehenkilöiden jako koe- ja kontrolliryhmään ei välttämättä ole mielekästä, koska kahden eri ihmisen tulokset eivät välttämättä ole vertailukelpoisia. Esimerkiksi, jos

mitataan muistia, niin kahdella ihmisellä voi olla lähtökohtaisesti hyvin erilainen tulos muistitestissä. Tällöin käsittelyn oikeaa vaikutusta ei välttämättä huomata, koska se peittyy ihmisten välisten erojen alle. Jos eri ihmisten tuloksia ei voida verrata keskenään, ratkaisuna on tehdä samalle henkilölle sekä koe- että kontrollikäsittely ja verrata näitä keskenään.

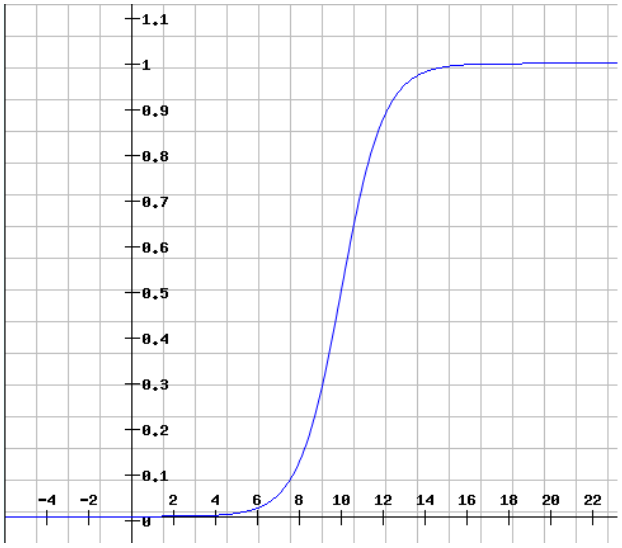
Kokeellisessa tutkimuksessa tärkeässä osassa on koesuunnittelu. Koesuunnittelun tarkoituksena on luoda tutkimuksesta mahdollisimman tehokas ja taloudellinen. Yleensä koesuunnittelun avulla pyritään valitsemaan optimaalisin koehenkilöiden (tai koeyksiköiden) määrä, käsittelyn suuruus, kokeen kesto tai mitattavien muuttujien määrä. Optimaalisin tilanne on yleensä se, että tuloksista saadaan suurin mahdollinen hyöty, ilman että kustannukset nousevat liian suuriksi.

Tässä tutkielmassa yhdistyy kaikki edellä oleva siten, että kognitiivisen neuropsykologian alueena on ilmiö, jota kutsutaan äänteiden kategoriseksi havaitsemiseksi. Tätä ilmiötä tutkitaan kokeellisella tutkimuksella, jossa jokainen koehenkilö muodostaa itse oman koe- ja kontrolliryhmänsä. Lisäksi koesuunnittelun osa-alueeseen kuuluu muuntuva koeasetelma, joka pyrkii mukautumaan yksilöllisesti jokaisen koehenkilön tarpeisiin, jolloin tutkimus on optimaalinen jokaisen koehenkilön osalta. Ennen kuin perehdytään tarkemmin kategoriseen havaitsemiseen tai muuntuvaan koeasetelmaan, lähdetään liikkeelle yksinkertaisemmasta esimerkkitilanteesta, joka on periaatteeltaan samanlainen kuin tämän tutkielman aihe.

Ajatellaan, että halutaan tarkastella ihmisen kuulokynnystä. Kuulokynnys on alhaisin äänenpainetaso, joka riittää aiheuttamaan havainnon. Tätä äänenpainetasoa mitataan äänenpainedesibeleinä eli dB SPL (*Sound Pressure Level*), tästä eteenpäin mittayksiköstä puhutaan lyhyesti desibelinä (dB). Ihmisen yleisenä kuulokynnyksenä pidetään 0 dB eli tätä pienempää ääntä yksikään ihminen ei enää pysty kuulemaan. (Jeans, 1968). On kuitenkin huomattava, että on olemassa ääniä, joiden äänenpainetaso on alle 0 desibeliä. Nämä äänet ovat kuulokynnyksen määritelmän mukaan vain liian hiljaisia, jotta ihminen ne kuulisi.

Huolimatta siitä, että kuulokynnykselle on olemassa yleinen määritelmä, se on kuitenkin jokaisella ihmisellä yksilöllinen eli sille ei voida määrittää sellaista desibeliarvoa, joka pätisi kaikille ihmisille. Kuulokynnystä voidaan kuitenkin mitata toistamalla ihmiselle ääniä eri äänenvoimakkuuksilla. Tällä menetelmällä voidaan määritellä, monenko desibelin kohdalla kyseisen ihmisen kuulokynnys on. Jos näistä mittauksista piirrettäisiin kuva, jossa x-akselilla olisi äänenvoimakkuus desibeleinä ja y-akselilla olisi, montako prosenttia äänistä on kuultu, saataisiin jotain kuvan 1 kaltaista.

Esimerkkikuvassa kyseinen henkilö ei siis ole kuullut yhtään ääntä, jonka äänenvoimakkuus on ollut 4 dB tai vähemmän ja puolestaan kuullut kaikki äänet, joiden äänenvoimakkuus on ollut 15 dB tai enemmän. Jos kuulokynnystä haluttaisiin tutkia, niin mielenkiinnon kohteena olisi esimerkkikuvan tapauksessa väli 4-15 dB, jossa henkilö on kuullut osan äänistä, mutta ei kaikkia.



Kuva 1:Esimerkkikuva kuulokynnyksestä, jossa x-akselilla on äänenvoimakkuus desibeleinä ja y-akselilla on kuultujen äänien määrä prosentteina.

Ongelmaksi kuulokynnyksen tutkimisessa nousee nyt se, että kuulokynnyksen tutkimista varten jokaisen tutkittavan henkilön kuulokynnys ja kuulokynnyksen kuvaaja täytyy määritellä erikseen. Sama ongelma on muiden vastaavanlaisten yksilöllisten ominaisuuksien, kuten äänteiden kategorisen havaitsemisen, suhteen.

Tästä päästään takaisin tutkielman aiheeseen eli äänteiden kategorisen havaitsemisen tutkimiseen. Äänteiden kategorisella havaitsemisella tarkoitetaan ilmiötä, jossa ihminen luokittelee eri äänteitä eri kategorioihin niiden akustisten ominaisuuksien perusteella. Tällöin kahden eri äänteen välimuotoa ei tunnisteta välimuodoksi, vaan se kuulostaa henkilön mielestä jommaltakummalta ääripään äänteeltä. Jokainen yksittäinen äänne muodostaa siis oman kategoriansa ja kaikki mahdolliset välimuodot luokitellaan kuuluviksi näihin kategorioihin. Voidaan siis ajatella, että mikäli kahden äänteen välille muodostetaan jatkumo äänteestä toiseen, pystytään löytämään jokin kynnysarvo, jota ennen äänne tunnistetaan ensimmäiseksi äänteeksi ja jonka jälkeen äänne tunnistetaan toiseksi äänteeksi. Kynnysarvon kohdalla ja sen lähiympäristössä äänteen luokittelu on epävarmempaa kuin ääripäiden lähellä. Mikäli henkilölle toistetaan kynnysarvon kohdalta äännettä useamman kerran, esimerkiksi kuudesti, hän yleensä luokittelee äänteen noin kolmella toistokerroista toiseksi äänteeksi ja lopuilla toistokerroilla toiseksi äänteeksi. Ääripäitä kohti mentäessä henkilö puolestaan luokittelee äänteen varmemmin vain toiseen kategoriaan kuuluvaksi.

Kuten kuulokynnyksen tapauksessa, kategorisen havaitsemisen kynnysarvo on jokaiselle ihmiselle yksilöllinen ja tästä syystä se on määriteltävä erikseen jokaisen henkilön kohdalla. Tämän kynnysarvon määrittelemiseksi käytettävän menetelmän tulisi olla mahdollisimman tarkka, nopea ja luotettava. Tähän tarkoitukseen on suunniteltu mukautuva malli, joka pyrkii valitsemaan kynnysarvon mahdollisimman tarkasti ja nopeasti. Tämän jälkeen voidaan keskittyä mittaamaan

tarkemmin kynnysarvoa ja sen lähellä olevia arvoja. Tärkeä osa mukautuvaa mallia on niin kutsuttu binäärinen etsintäalgoritmi, joka pyrkii puolitushaulla löytämään mahdollisimman nopeasti kynnysarvon, jossa sama äänne luokitellaan kumpaankin kategoriaan.

Etsintäalgoritmin tapauksessa ajatellaan, että selittäjä x on soitettava ääni. Aiemmassa kuulokynnysesimerkissä selittäjän arvot olivat desibelejä. Vaste y puolestaan kertoo, kumpaan kahdesta kategoriasta kuultu äänne luokitellaan (0 tai 1). Esimerkissä vaste sai arvoja ”ei kuullut” (0) ja ”kuuli” (1). Kynnysarvo on tällöin siinä kohdassa, jossa vaste y saa sekä arvoja nolla että yksi, esimerkissä se oli siis 4-15 dB. Estimoinnissa käytetään yleensä joko logit-, probit- tai komplementaarista log-log-mallia. Tässä tutkielmassa käytössä on logit-malli.

Tässä tutkielmassa on toteutettu kategorisen havaitsemisen tutkimiseen tarkoitettu MATLAB-ohjelma. Binäärisen etsintäalgoritmin pohjana on käytetty Juha Karvasen optdesign-pakettia (Karvanen, 2008) ja komentoikkunaan tulostamisessa on käytetty Altmanin cprintf-ohjelmaa (Altman, 2012), muilta osin MATLAB-koodi on minun tekemäni. Ohjelma toistaa koehenkilöille ääniä ja rekisteröi vastaukset. Vastausten perusteella binäärinen etsintäalgoritmi laskee, minkä äänen kohdalta kynnysarvoa seuraavaksi etsitään. Kun kynnysarvo on löytynyt, binääristä etsintäalgoritmia ei enää tarvita, vaan ääniä toistetaan satunnaisessa järjestyksessä, painottaen etsintäalgoritmin löytämää kynnysarvoa sekä sen lähellä olevia arvoja. Soitettujen äänten järjestys tallennetaan ja äänet soitetaan samassa järjestyksessä uudelleen koehenkilölle, jonka tarvitsee enää pelkästään kuunnella ääniä. Koko tutkimuksen ajan koehenkilöiltä tallennetaan aivosähkökäyrää eli EEG:tä myöhempää analysointia varten.

Pro gradu -tutkielmaa varten suoritettiin 5 päivänä mittauksia, joihin osallistui yhteensä 18 koehenkilöä. Minä olin vastuussa koehenkilöiden keräämisestä, taustatietojen kyselemisestä sekä ohjelman toiminnan valvomisesta. Mittauksissa apuna oli psykologian laitokselta Suvi Karla, joka vastasi pääosin aivosähkökäyrän mittaamisesta (muun muassa EEG-myssyjen laittamisesta koehenkilöille, mittausten tallentamisesta ja välineiden puhdistuksesta). Parilta koehenkilöltä sain minä hoitaa EEG-mittaukset Suvin valvonnassa ja kolmelta koehenkilöltä hoidin mittaukset kokonaan yksin.

Tutkielman rakenne on seuraava, luvussa 2 käsitellään binääriseen vasteeseen liittyvää teoriaa eli logistista mallia, D-optimaalisuutta, suurimman uskottavuuden estimaattien olemassaoloa ja binääristä etsintäalgoritmia. Luvussa 3 puolestaan esitellään varsinainen sovelluskohde eli äänteiden kategorisen havaitsemisen tutkiminen ja siihen liittyviä käsitteitä. Luvussa 4 esitellään yksityiskohtaisesti ohjelma, joka on tehty äänteiden kategorisen havaitsemisen tutkimista varten. Luku 5 sisältää kuvauksen mittauksista sekä ohjelman toiminnan analysointia. Lopuksi luku 6 sisältää yhteenvedon sekä pohdintaa ohjelman hyödyllisyydestä ja käyttökelpoisuudesta.

2 Binäärisen vasteen teoriaa

Vastetta, joka voi saada kahta eri arvoa, kutsutaan binääriseksi (kaksiarvoiseksi). Näihin kahteen arvoon viitataan yleensä ”onnistumisena” ja ”epäonnistumisena” tai vastaavasti arvoilla 1 ja 0. Suoritetaan koe, jossa kokeen tekijä kontrolloi selittäjää ja onnistumisen todennäköisyyttä mallintaa selittäjän suhteen monotoninen funktio. Tällöin selittäjää vastaaviin vasteen arvoihin voidaan sovittaa käyrä, jota kutsutaan vastekäyräksi. Tämän vastekäyrän muoto tunnetaan, mutta sen sijainnin ja kulmakertoimen määrittäviä parametreja puolestaan ei tunneta. Nämä tuntemattomat parametrit halutaan estimoida, mutta niihin liittyvä ennakkotieto (prioritieto) on vähäistä. Lähestymistapana on valita jokin sellainen parametrinen malli, jota käytetään mallintamaan kaksiarvoista vastetta. Yleisesti käytettyjä malleja on kolme: logistiseen jakaumaan perustuva logit-malli, normaalijakaumaan perustuva probit-malli ja Gompertz-jakaumaan perustuva komplementaarinen log-log-malli. Kaikkien näiden mallien tarkoituksena on mallintaa binäärisen vasteen Y ja selittäjän x välistä riippuvuutta. Nämä mallit voidaan esittää yleistettynä lineaarisena mallina

$$P(Y = 1) = E(Y) = F(\beta_0 + \beta_1 x), \tag{2.1}$$

missä vastekäyrä F on kertymäfunktio ja β_0 ja β_1 ovat estimoitavan mallin parametrit. Tässä tutkielmassa tarkastellaan logit-mallia, jolloin kyseessä on logistinen regressiomalli.

Luvussa 2.1 esitellään logistinen regressiomalli, mallin johtaminen sekä parametrien estimointi. Luvussa 2.2 kerrotaan optimaalisista malleista logistisessa regressiossa (erityisesti D-optimaalisuudesta) ja luvussa 2.3 tarkastellaan, milloin suurimman uskottavuuden estimaatit ovat olemassa binäärisen logistisen regressiomallin tapauksessa. Viimeisessä luvussa esitellään binäärinen etsintäalgoritmi.

2.1 Logistinen regressiomalli

Tämä luku perustuu artikkeliin Czepiel (2002). Logistista regressiota käytetään mallintamaan kategorista, yleensä kaksiarvoista, vastemuuttujaa Y. Logistinen regressiomalli kuuluu yleistettyihin lineaarisiin malleihin, joita käytetään mallintamaan riippuvuutta vastemuuttujan Y ja selittäjän x välillä. Lineaarinen regressiomalli on erikoistapaus yleistetyistä lineaarisista malleista ja yleisesti käytetty, mutta se ei sovi kategorisille vastemuuttujille, koska vasteen arvot eivät ole suhdeasteikollisia ja lisäksi virhetermit eivät ole normaalijakautuneita.

Lineaarisissa regressiomalleissa halutaan mallintaa vastemuuttujan lineaarista riippuvuutta selittävistä muuttujista, jolloin riippuvan muuttujan odotusarvo on suhteessa riippumattomien muuttujien lineaarikombinaatioon ja riippumattomien muuttujien parametreihin. Yleistetyissä lineaarisissa malleissa puolestaan halutaan mallintaa tapahtumien todennäköisyyttä, jolloin lineaariset komponentit ovat suhteessa riippuvan muuttujan tulosten todennäköisyysfunktioon. Logistisessa regressiossa tämä todennäköisyysfunktio on logit-muunnos, joka saadaan, kun otetaan luonnollinen logaritmi tapahtuman todennäköisyyksien suhteesta. Lineaarisessa regressiossa parametrit estimoidaan käyttämällä pienimmän neliösumman menetelmää (PNS), kun taas logistiseen regressiomalliin tämä menetelmä ei sovi, koska se ei pysty tuottamaan harhattomia estimaattoreita. Tästä syystä logistisessa regressiossa käytetään pienimmän neliösumman menetelmän sijaan suurimman uskottavuuden menetelmää, jolla ratkaistaan parhaiten aineistoon sopivat parametrit. Seuraavaksi määritellään tarkemmin logistinen regressiomalli ja näytetään, kuinka parametrien estimointi tapahtuu suurimman uskottavuuden menetelmällä.

2.1.1 Mallin johtaminen

Olkoon Z kaksiarvoinen (binäärinen) satunnaismuuttuja ja on olemassa aineisto, jonka otoskoko on M. Ajatellaan, että havainnot laitetaan allekkain yhteen sarakkeeseen ja viereiseen sarakkeeseen laitetaan jokaista havaintoa vastaava vasteen arvo Z. Mikäli jokainen havainto on riippumaton eli havainnon arvot eivät riipu muista havainnoista, voidaan ajatella, että $\boldsymbol{Z}$ on sarakevektori, jossa on M kappaletta binomisia satunnaismuuttujia Z_i. Merkitään ”onnistumista” arvolla 1 ja ”epäonnistumista” arvolla 0 ja ryhmitellään aineisto siten, että jokainen rivi vastaa yhtä muuttujien arvojen yhdistelmää. Näitä rivejä kutsutaan luokiksi. Merkitään muuttujalla N luokkien kokonaislukumäärää ja merkitään muuttujalla $\boldsymbol{n}$ sarakevektoria, jonka alkiot n_i kuvaavat havaintojen määrää luokassa i, missä $i = 1, \dots, N$ ja $\sum_{i=1}^{N} n_i = M$. Koska vaste Z_i on binominen, se noudattaa Bernoulli-jakaumaa eli $P(Z_i = z_i) = {\pi_i}^{z_i}(1 - \pi_i)^{1-z_i}$, missä $z_i = 0{,}1$. Muuttuja π_i kuvaa onnistumisen todennäköisyyttä mille tahansa havainnolle i.nnessä luokassa (eli $\pi_i = P(Z_i = 1|i)$). Kaikista todennäköisyyksistä π_i voidaan muodostaa sarakevektori $\boldsymbol{\pi}$, jonka pituus on N.

Olkoon $\boldsymbol{Y}$ sarakevektori, jonka pituus on N ja jokainen alkio Y_i on satunnaismuuttuja, joka kuvaa onnistumisten määrää Z luokalle i. Sarakevektorin $\boldsymbol{y}$ alkiot y_i kuvaavat havaittujen onnistumisten lukumäärää jokaiselle luokalle. Tällöin muuttuja Y_i noudattaa binomijakaumaa eli $P(Y_i = y_i) = \binom{n_i}{y_i} {\pi_i}^{y_i}(1 - \pi_i)^{n-y_i}$, missä $y_i = 0, \dots, n_i$. Logit-muunnokseen tarvitaan ristitulosuhdetta (*odds*), joka on onnistumisen todennäköisyys suhteessa epäonnistumisen todennäköisyyteen eli

$$\text{odds}_i = \frac{\pi_i}{1-\pi_i}. \tag{2.1.1}$$

Logit-muunnos (tai logit-linkki) saadaan, kun otetaan ristitulosuhteesta luonnollinen logaritmi eli

$$\text{logit}(\pi_i) = \log\frac{\pi_i}{1-\pi_i}. \tag{2.1.2}$$

Seuraavaksi määritellään lineaarinen komponentti, joka kertoo, miten todennäköisyydet π_i riippuvat selittäjistä x_i. Tämä lineaarinen komponentti sisältää asetelmamatriisin $\boldsymbol{X}_{N\times(K+1)}$ (design matrix), jossa on selittäjien arvot x_i sekä sarakevektorin $\boldsymbol{\beta}_{(K+1)\times 1}$, jossa on estimoitavat parametrit β_k. Asetelmamatriisissa on K kappaletta riippumattomia selittäjiä, joiden määrä on määritelty mallissa ja jokaisen asetelmamatriisin rivin ensimmäinen alkio $x_{i0} = 1$. Parametrivektorissa puolestaan on yksi parametri jokaista asetelmamatriisin saraketta K kohden ja lisäksi yksi parametri, β_0, leikkauspisteelle.

Logistinen regressiomalli saadaan, kun määritellään linkkifunktion ja lineaarisen komponentin yhteys seuraavasti:

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = \sum_{k=0}^{K} x_{ik}\beta_k \,, i = 1,2,\dots,N. \tag{2.1.3}$$

Tästä yhtälöstä voidaan ratkaista havainnon onnistumisen todennäköisyys π_i korottamalla kumpikin puoli luvun e potenssiin, jolloin yhtälö saadaan seuraavaan muotoon

$$\pi_i = \left(\frac{e^{\sum_{k=0}^{K} x_{ik}\beta_k}}{1 + e^{\sum_{k=0}^{K} x_{ik}\beta_k}}\right). \tag{2.1.4}$$

Muotoa (2.1.4) tarvitaan seuraavassa luvussa uskottavuusyhtälön muodostamiseen.

2.1.2 Parametrien estimointi

Logistisessa regressiossa halutaan estimoida tuntemattomat parametrivektorin $\boldsymbol{\beta}$ arvot. Tämä tapahtuu suurimman uskottavuuden menetelmällä, jossa tarkoituksena on löytää sellaiset parametrien β_k estimaatit $\hat{\beta}_k$, joilla pystytään kaikkein todennäköisimmin generoimaan havaitussa otoksessa olevat havainnot. Suurimman uskottavuuden estimaatit johdetaan niin sanotusta uskottavuusyhtälöstä, joka johdetaan riippuvan muuttujan (tässä tapauksessa Y_i) todennäköisyysjakaumasta. Uskottavuusyhtälö samanlainen kuin vektorin $\boldsymbol{Y}$ yhteistiheysfunktio, joka saadaan muodostettua jakaumien tulona. Erona on, että suurimman uskottavuuden menetelmässä ajatellaan, että havainnot $\boldsymbol{y}$ ovat kiinteitä ja parametrit $\boldsymbol{\beta}$ ovat muuttujia, kun taas yhteistiheysfunktiossa vektori $\boldsymbol{y}$ on tuntematon ja parametrit $\boldsymbol{\beta}$

tunnettuja. Toisin sanoen muuttujat ovat siis käänteiset uskottavuusfunktiossa ja yhteistiheysfunktiossa. Binomijakauman tapauksessa uskottavuusfunktio L voidaan kirjoittaa seuraavasti:

$$L(\boldsymbol{\beta}|\boldsymbol{y}) = f(\boldsymbol{y}|\boldsymbol{\beta}) = \prod_{i=1}^{N} \binom{n_i}{y_i} \pi_i^{y_i} (1-\pi_i)^{n_i - y_i}. \tag{2.1.5}$$

Tapoja, joilla voidaan järjestää onnistumisten lukumäärä y_i yrityksissä n_i, on $\binom{n_i}{y_i}$ erilaista. Koska onnistumisen todennäköisyys missä tahansa yrityksessä n_i on π_i, todennäköisyys onnistua y_i kertaa on $\pi_i^{y_i}$. Vastaavasti todennäköisyys epäonnistua $n_i - y_i$ kertaa on $(1-\pi_i)^{n_i - y_i}$.

Suurimman uskottavuuden estimaatit ovat ne arvot β, jotka maksimoivat uskottavuusfunktion L (2.1.5). Seuraava askel on siis maksimoida tämä funktio. Maksimointi tapahtuu derivoimalla parametrien $\boldsymbol{\beta}$ suhteen: maksimi saadaan, kun ensimmäinen derivaatta asetetaan nollaksi. Tämä piste on todella maksimi, mikäli toinen derivaatta on pienempi kuin nolla. Uskottavuusyhtälöä voidaan yksinkertaistaa ottamalla siitä logaritmi. Koska logaritmi on monotoninen funktio, uskottavuusfunktion maksimi on myös logistisen uskottavuusfunktion maksimi ja toisin päin. Logistinen uskottavuusfunktio on muotoa

$$l(\boldsymbol{\beta}|\boldsymbol{y}) = \log L\,(\boldsymbol{\beta}|\boldsymbol{y}) = \sum_{i=1}^{N} \left(\log \binom{n_i}{y_i} + y_i \log(\pi_i) + (n_i - y_i) \log(1-\pi_i) \right) \tag{2.1.6}$$

Käyttämällä logaritmin laskusääntöä $\log(x/y) = \log(x) - \log(y)$, voidaan yhtälö (2.1.6) kirjoittaa muodossa

$$l(\boldsymbol{\beta}|\boldsymbol{y}) = \sum_{i=1}^{N} \left(\log \binom{n_i}{y_i} + y_i \log\left(\frac{\pi_i}{1-\pi_i}\right) + n_i \log(1-\pi_i) \right) \tag{2.1.7}$$

Käytetään seuraavaksi hyväksi yhtälössä 2.1.4 esitettyä muotoa logit-linkille, jolloin saadaan korvattua todennäköisyydet π_i. Lisäksi käytetään uudelleen edellä mainittua logaritmien laskusääntöä, jolloin yhtälö 2.1.7 saadaan seuraavaan muotoon

$$\begin{aligned} l(\boldsymbol{\beta}|\boldsymbol{y}) &= \sum_{i=1}^{N} \left(\log \binom{n_i}{y_i} + y_i \log\left(e^{\sum_{k=0}^{K} x_{ik}\beta_k}\right) + n_i \log\left(\frac{1}{1 + e^{\sum_{k=0}^{K} x_{ik}\beta_k}}\right) \right) \\ &= \sum_{i=1}^{N} \left(\log \binom{n_i}{y_i} + y_i \sum_{k=0}^{K} x_{ik}\beta_k - n_i \log\left(1 + e^{\sum_{k=0}^{K} x_{ik}\beta_k}\right) \right). \end{aligned} \tag{2.1.8}$$

Seuraavaksi derivoidaan logaritminen uskottavuusfunktio (2.1.8) ensimmäisen kerran. Voidaan huomata, että

$$\frac{d}{d\beta_k}\sum_{k=0}^{K} x_{ik}\beta_k = x_{ik}\,, \tag{2.1.9}$$

sillä muut summan termit eivät riipu parametrista β_k ja niitä voidaan kohdella vakioina. Tämän lisäksi käytetään derivointisääntöä $\frac{d}{dy}\log y = \frac{1}{y}$. Näin ollen derivointi tuottaa seuraavan yhtälön:

$$\begin{aligned}\frac{d\, l(\boldsymbol{\beta}|\boldsymbol{y})}{d\beta_k} &= \sum_{i=1}^{N}\left(y_i\, x_{ik} - n_i \frac{e^{\sum_{k=0}^{K} x_{ik}\beta_k}}{1+e^{\sum_{k=0}^{K} x_{ik}\beta_k}} x_{ik}\right) \\ &= \sum_{i=1}^{N} (y_i x_{ik} - n_i \pi_i x_{ik})\end{aligned} \tag{2.1.10}$$

Uskottavuusyhtälön maksimi löydetään, kun asetetaan derivaatta jokaisen parametrin β_k suhteen nollaksi yhtälössä (2.1.10). Parametrivektorin $\boldsymbol{\beta}$ suurimman uskottavuuden estimaatit löydetään, kun asetetaan jokainen $K+1$ yhtälöstä nollaksi yhtälössä (2.1.10) ja ratkaistaan jokainen β_k. Ratkaisu on maksimi, jos toisen osittaisen derivaatan matriisi on negatiivisesti definiitti (jokainen alkio matriisin diagonaalilla on pienempi kuin nolla). Derivoimalla jokainen $K+1$ yhtälöstä yhtälössä (2.1.10) toisen kerran jokaisen parametrin β_k suhteen saadaan muodostettua parametrien estimaateille varianssi-kovarianssimatriisi. Matriisin yleinen muoto on

$$\begin{aligned}\frac{d}{d\beta_k}\sum_{i=1}^{N}(y_i x_i - n_i \pi_i x_{ik}) &= \frac{d}{d\beta_k}\sum_{i=1}^{N} -n_i x_{ik}\pi_i \\ &= -\sum_{i=1}^{N} n_i x_{ik}\frac{d}{d\beta_k}\pi_i.\end{aligned} \tag{2.1.11}$$

Lasketaan seuraavaksi derivointisääntöjen avulla, mitä todennäköisyyksien π_i derivaatta on

$$\frac{d}{d\beta_{k'}}\pi_i = \frac{d}{d\beta_{k'}}\left(\frac{e^{\sum_{k=0}^{K} x_{ik}\beta_k}}{1+e^{\sum_{k=0}^{K} x_{ik}\beta_k}}\right)$$

$$= \frac{\left(1+e^{\sum_{k=0}^{K} x_{ik}\beta_k}\right) e^{\sum_{k=0}^{K} x_{ik}\beta_k}\frac{d}{d\beta_{k'}}\sum_{k=0}^{K} x_{ik}\beta_k - e^{\sum_{k=0}^{K} x_{ik}\beta_k} e^{\sum_{k=0}^{K} x_{ik}\beta_k}\frac{d}{d\beta_{k'}}\sum_{k=0}^{K} x_{ik}\beta_k}{(1+e^{\sum_{k=0}^{K} x_{ik}\beta_k})^2}$$

$$= \frac{\left(1 + e^{\sum_{k=0}^{K} x_{ik}\beta_k} - e^{\sum_{k=0}^{K} x_{ik}\beta_k}\right) e^{\sum_{k=0}^{K} x_{ik}\beta_k} \frac{d}{d\beta_{k'}} \sum_{k=0}^{K} x_{ik}\beta_k}{\left(1 + e^{\sum_{k=0}^{K} x_{ik}\beta_k}\right)^2}$$

$$= \frac{e^{\sum_{k=0}^{K} x_{ik}\beta_k} \frac{d}{d\beta_{k'}} \sum_{k=0}^{K} x_{ik}\beta_k}{\left(1 + e^{\sum_{k=0}^{K} x_{ik}\beta_k}\right)^2}$$

$$= \frac{e^{\sum_{k=0}^{K} x_{ik}\beta_k}}{(1 + e^{\sum_{k=0}^{K} x_{ik}\beta_k})} \frac{1}{(1 + e^{\sum_{k=0}^{K} x_{ik}\beta_k})} x_{ik}. \quad (2.1.12)$$

Käytetään seuraavaksi jälleen kerran todennäköisyyksiä π_i avuksi ja sijoitetaan tulos (2.1.12) kaavaan (2.1.11), jolloin logaritmisen uskottavuusfunktion toinen derivaatta voidaan kirjoittaa seuraavasti

$$\frac{d^2 l(\boldsymbol{\beta}|\boldsymbol{y})}{d\beta_k d\beta_{k'}} = -\sum_{i=1}^{N} n_i x_{ik} \pi_i (1 - \pi_i) x_{ik'}. \quad (2.1.13)$$

Odotettu informaatiomatriisi, toiselta nimeltään Fisherin informaatiomatriisi $\boldsymbol{J}$, on sama kuin kaavassa (2.1.13) oleva toinen derivaatta (Monahan, 2001).

Kun asetetaan ensimmäisen derivaatan (2.1.10) yhtälöt nollaksi, saadaan $K + 1$ yhtälön epälineaarinen yhtälöryhmä, jossa jokaisessa on $K + 1$ tuntematonta muuttujaa. Tämän yhtälöryhmän ratkaisu on vektori, jossa on alkiot $\hat{\beta}_k$. Tämä yhtälöryhmä on mahdoton ratkaista algebrallisesti, joten ainut vaihtoehto on estimoida ratkaisu numeerisesti jollakin iteratiivisella menetelmällä. (Czepiel, 2002.)

2.2 Optimaaliset mallit ja *D*-optimaalisuus

2.2.1 Yleisesti

Optimaalisen koesuunnittelun teorian tarkoituksena on yrittää estimoida parametrit β_0 ja β_1 mahdollisimman tehokkaasti. Käytännössä tämä tarkoittaa sitä, että tutkitaan, miten prediktoritasot tulisi koesuunnittelussa valita, jotta parametrien estimointi halutulla tarkkuudella onnistuisi mahdollisimman pienellä otoskoolla. *D*-optimaalisen mallin nimi tulee siitä, että siinä maksimoidaan mallin informaatiomatriisin determinanttia (D = determinantti).

Usean selittäjän lineaarinen regressiomalli voidaan kirjoittaa muodossa

$$E(y) = \mu = \eta = \boldsymbol{\beta}^T f(x). \quad (2.2.1)$$

Voidaan huomata, että parametrin y keskiarvo μ ja lineaarinen prediktori η ovat samat. Lineaarinen malli määrittelee nimensä mukaisesti vasteen ja prediktorin välisen suhteen lineaariseksi, kun taas yleistetyissä lineaarisissa malleissa vasteen ja prediktorin suhde ei ole lineaarinen, vaan se määräytyy niin kutsutun linkkifunktion $g(\mu) = \eta$ perusteella. Linkkifunktio $g(\mu)$ on identiteettifunktio tai identtinen kuvaus, sillä se kuvaa jokaisen lähtöjoukon alkion itsekseen eli $\mu = \eta$. Logistinen linkkifunktio on

$$\eta = \log\left(\frac{\mu}{1-\mu}\right). \tag{2.2.2}$$

Mikäli ”onnistuminen” ($y = 1$) korvataan ”epäonnistumisella” ($y = 0$), malli pysyy samana, mutta sen etumerkki vaihtuu. (Atkinson et al. 2007.)

Optimaalinen malli esitetään mittana θ, joka saa arvoja yli joukon X. Mitta θ voidaan kirjoittaa seuraavasti

$$\theta = \begin{bmatrix} x_1 & x_2 \dots x_n \\ q_1 & q_2 \dots q_n \end{bmatrix}. \tag{2.2.3}$$

Tällöin kokeita suoritetaan joukon X arvoilla $x_1, \dots, x_n$. Näitä ensimmäisellä rivillä olevia joukon X arvoja kutsutaan asetelmapisteiksi (*design points*) ja ne sisältävät faktoreiden arvot. Toisella rivillä olevat arvot q_i puolestaan antavat asetelmapisteisiin liittyvät asetelmapainot (*design weights*). Yleensä D-optimaalisella p-parametrisella mallilla asetelman parametreilla on kaikilla sama paino $1/p$. Koska θ on mitta, sen integraali yli joukon X on 1 ($\int_X \theta(dx) = 1$). Luonnollisesti painoille q_i pätee, että ne saavat arvoja väliltä [0,1], koska painojen summan on oltava 1. (Atkinson et al. 2007.)

Parametrien määrä on siis p ja N on mittausten yhteismäärä asetelmapisteissä. Yleistetyille lineaarisille malleille asymptoottinen kovarianssimatriisi on muotoa $\boldsymbol{R}^T\boldsymbol{Q}\boldsymbol{R}$, missä $\boldsymbol{R}$ on kokoa $N \times p$ oleva asetelmamatriisi. Diagonaalimatriisissa $\boldsymbol{Q}$ puolestaan ovat mallin painot ja luonnollisesti se on kokoa $N \times N$ oleva matriisi. Painot riippuvat kolmesta asiasta: lineaarisen mallin parametreista (tuntemattomia), virheen jakaumasta sekä näiden välisestä linkkifunktiosta. Koska mallin parametreja ei tunneta, joudutaan asetelmaa varten käyttämään ennakko-oletusta (priorioletusta) parametrien jakaumasta. (Atkinson et al. 2007.)

Yleistettyjen lineaaristen mallien informaatiomatriisi on painotetussa muodossa, koska parametrien β_i suurimman uskottavuuden estimaattori on sievennetyssä muodossa sama kuin painotettu pienimmän neliösumman estimaattori. Yksittäisen havainnon painot saadaan kaavasta

$$w = V^{-1}(\mu)\left(\frac{d\mu}{d\eta}\right)^2, \tag{2.2.4}$$

missä V^{-1} on varianssifunktion $V(\mu) = \mu(1-\mu)$ käänteisfunktio. Painot riippuvat siis sekä parametrin y jakaumasta että linkkifunktiosta.

Yhden selittäjän logistinen regressiomalli voidaan ilmasta muodossa

$$\log\left(\frac{\mu}{1-\mu}\right) = \eta = \beta_0 + \beta_1 x. \tag{2.2.5}$$

Jotta saadaan mallin painojen matriisi $\boldsymbol{Q}$ laskettua, derivoidaan logistinen linkki

$$\frac{d\eta}{d\mu} = \frac{1}{\mu(1-\mu)} \tag{2.2.6}$$

ja yhdistetään saatu tulos varianssifunktion $V(\mu)$ kanssa. Näin saadaan painoille yksinkertainen muoto

$$\boldsymbol{Q} = \mu(1-\mu). \tag{2.2.7}$$

(Atkinson et al. 2007.)

Yleisesti optimaalisten mallien tarkoituksena on minimoida jokin epätarkkuuden mitta Ψ. Tämä tapahtuu D-optimaalisuuden tapauksessa Fisherin informaatiomatriisin kautta. Olkoon Fisherin informaatiomatriisi $\boldsymbol{J}$, suhteessa asetelmamatriisiin $\boldsymbol{R}$ seuraavasti

$$\boldsymbol{J} = \boldsymbol{R}^T\boldsymbol{R}. \tag{2.2.8}$$

Mikäli malli on jatkuva, havainnot ovat riippumattomia ja $q(x_i)$ joukko tunnettuja painoja ($\boldsymbol{Q} = \mathrm{diag}(q(x_i))$) niin tällöin informaatiomatriisi saa muodon

$$\boldsymbol{J}(q,\theta) = \int q(x) f(x) f^T(x) \theta(dx), \tag{2.2.9}$$

missä $f^T(x_i)$ on asetelmamatriisin $\boldsymbol{R}$ i. rivi. Näin ollen *D*-optimaalisuus määritellään seuraavasti

$$\Psi\big(\boldsymbol{J}(\theta)\big) = \log|\boldsymbol{J}^{-1}(\theta)| = -\log|\boldsymbol{J}(\theta)| \tag{2.2.10}$$

Tarkoituksena on informaatiomatriisin $\boldsymbol{J}(\theta)$ determinantin maksimoiminen. Globaali minimi löydetään silloin, kun determinantista otetaan logaritmi. *D*-optimaalinen malli θ^* siis maksimoi determinantin $|\boldsymbol{J}(\theta)|$ tai vastaavasti minimoi determinantin $|\boldsymbol{J}^{-1}(\theta)|$. Determinantin $|\boldsymbol{J}(\theta)|$ maksimointi on sama asia, kuin determinantin $|\boldsymbol{R}|^2$ tai $|\boldsymbol{R}|$ maksimointi (Syed ym. 2011). Joskus voidaan myös käyttää konveksia optimointia maksimoimalla $\log|\boldsymbol{J}(\theta)|$ tai minimoimalla $-\log|\boldsymbol{J}(\theta)|$. *D*-optimaalinen malli on näin ollen asetelmamatriisi, jonka determinantti on maksimoitu. (Atkinson et al. 2007.)

Matemaattisesti *D*-optimaalinen malli määritellään seuraavasti: olkoon $\bar{\boldsymbol{J}} \in \mathbb{R}^{M\times M}$ joukko neliömatriiseja ja $\bar{\boldsymbol{J}} = \{\boldsymbol{J}: [J]_{i,j} \in \{0,1\}\ \forall\, i,j = 1,\dots,M\}$, missä M on faktoreiden määrä sekä $[J]_{i,j}$ on informaatiomatriisin i.nnen rivin ja j.nnen sarakkeen alkio. Tällöin matriisia $\boldsymbol{R} \in \bar{\boldsymbol{J}}$ kutsutaan asteen M optimaaliseksi malliksi, jos $|\boldsymbol{R}| \geq |\boldsymbol{J}|$ kaikilla $\boldsymbol{J} \in \bar{\boldsymbol{J}}$. (Syed ym. 2011.)

2.2.2 *D*-optimaalinen malli logistiselle regressiolle

Suurimman uskottavuuden menetelmällä yritetään löytää sellainen suurimman uskottavuuden estimaattori, joka maksimoi uskottavuusfunktion arvon. Laskeminen tapahtuu uskottavuusfunktion logaritmin kautta. Logaritmisen uskottavuusfunktion maksimiarvot ovat tietyllä neliöllisellä alueella, jota kutsutaan uskottavuusalueeksi. Tätä uskottavuusaluetta voidaan approksimoida ellipsillä, jonka koko riippuu funktion parametrien suurimman uskottavuuden estimaattien informaatiomatriisista, jota kutsutaan myös Fisher-informaatiomatriisiksi. Kun Fisher-informaatiomatriisin determinantti maksimoidaan, niin uskottavuusalueen koko minimoituu. Fisher-informaatiomatriisin determinantista saadaan tällöin laskettua mallin *D*-optimaaliset tasot. Optimaaliset tasot kertovat, mitä selittäjän arvoja tarvitaan, jotta malli olisi optimaalinen. Seuraavaksi esitellään näiden *D*-optimaalisten tasojen laskeminen logistiselle regressiomallille.

Jos havaitaan n kappaletta binäärisiä vasteita $y_1, \dots, y_n$, vasteiden odotusarvot ilmaistuna selittäjän x avulla ovat

$$E(y_i) = [1 + \exp(-(\beta_0 + \beta_1 x_i))]^{-1}. \tag{2.2.11}$$

Halutaan estimoida sijaintiparametri β_0 ja skaalaparametri β_1, jota varten otoskoon on oltava riittävän suuri. Olkoon $\hat{\beta}_0$ ja $\hat{\beta}_1$ parametrien suurimman uskottavuuden estimaatit, tällöin uskottavuusfunktion maksimiarvot sisältävä uskottavuusalue on

$$A_{c,N}(\hat{\beta}_0, \hat{\beta}_1) = \left\{(\beta_0, \beta_1): \sum_{i=1}^{N} \left[l_i(\hat{\beta}_0, \hat{\beta}_1) - l_i(\beta_0, \beta_1)\right] \leq c\right\}, \tag{2.2.12}$$

missä $l_i(\beta_0, \beta_1) = y_i(\beta_0 + \beta_1 x_i) - \log[\exp(\beta_0 + \beta_1 x_i) + 1]$ ja luku c on vakio, joka määrittelee uskottavuusalueen koon. Uskottavuusaluetta A voidaan arvioida ellipsillä

$$\left\{(\beta_0, \beta_1): (\beta_0 - \hat{\beta}_0, \beta_1 - \hat{\beta}_1)\boldsymbol{J}(\hat{\beta}_0, \hat{\beta}_1)(\beta_0 - \hat{\beta}_0, \beta_1 - \hat{\beta}_1)^T \leq 2c\right\}, \tag{2.2.13}$$

missä Fisher-informaatiomatriisi $\boldsymbol{J}$ on symmetrinen matriisi

$$\boldsymbol{J}(\hat{\beta}_0, \hat{\beta}_1) = \begin{bmatrix} -\sum_{i=1}^{N} \frac{d^2 l_i(\beta_0, \beta_1)}{d{\beta_0}^2} & -\sum_{i=1}^{N} \frac{d^2 l_i(\beta_0, \beta_1)}{d\beta_0 d\beta_1} \\ -\sum_{i=1}^{N} \frac{d^2 l_i(\beta_0, \beta_1)}{d\beta_0 d\beta_1} & -\sum_{i=1}^{N} \frac{d^2 l_i(\beta_0, \beta_1)}{d{\beta_1}^2} \end{bmatrix}. \quad (2.2.14)$$

Lisäksi oletetaan, että on jotkin riittävän hyvät alustavat estimaatit parametreille $\hat{\beta}_0$ ja $\hat{\beta}_1$, jotta pystytään tarkasti ennustamaan informaatiomatriisin determinantti. (Minkin 1987.)

Seuraavaksi esitetään D-optimaalisten tasojen määrittäminen logistiselle regressiomallille, kuten Minkin (1987) on sen esittänyt. Aiemmin asiaa on tarkasteltu myös muun muassa artikkelissa Abdelbasit ja Plackett (1983). Lasketaan selittäjän x arvot $x_1, \dots, x_n$, jotka maksimoivat determinantin $|\boldsymbol{J}(\hat{\beta}_0, \hat{\beta}_1)|$. Optimaaliset arvot löydetään käyttämällä hyväksi tietoa, että determinantti $|\boldsymbol{J}(\hat{\beta}_0, \hat{\beta}_1)|$ voidaan kirjoittaa kahden termin erotuksena, joista kumpikaan ei ole negatiivinen eli

$${\beta_1}^2 |J(\beta_0, \beta_1)| = \sum_{i,j} \omega_i \, \omega_j \theta_j^2 - \left(\sum_i \omega_i \theta_i \right)^2, \quad (2.2.15)$$

missä $\theta_i = \beta_0 + \beta_1 x_i$ ja $\omega_i = \omega(\theta_i) = \frac{\exp(\theta_i)}{(1+\exp(\theta_i))^2}$. Tästä pystytään ratkaisemaan ne selittäjän arvot $x_1, \dots, x_n$, jotka tuottavat mahdollisimman tarkat estimaatit parametreille $\hat{\beta}_0$ ja $\hat{\beta}_1$ uskottavuusalueiden $A_{c,N}(\hat{\beta}_0, \hat{\beta}_1)$ minimoituessa. Tämä tapahtuu maksimoimalla kaavan (2.2.15) ensimmäinen termi. Optimaalinen malli riippuu parametrien $\hat{\beta}_0$ ja $\hat{\beta}_1$ arvoista, joten kirjoitetaan ensimmäinen termi kanonisessa muodossa $\beta_1 = 1$ ja $\beta_0 = 0$

$$\sum_{i,j} \frac{\exp(x_i)\exp(x_j){x_j}^2}{(1+\exp(x_i))^2(1+\exp(x_j))^2}. \quad (2.2.16)$$

Kaava (2.2.16) maksimoituu, kun $\theta_i = (\exp(\theta_i) + 1)/(\exp(\theta_i) - 1)$. Laskemalla nähdään, että ehto täyttyy silloin, kun $\theta_i \approx \pm 1.5434$.

Asetelmapisteet saadaan laskettua kanonisista muodoista $\eta = 1 \times x + 0 = x$ ja $\eta = 0 \times x + 1 = 1$. Tällöin ylempi asetelmapiste x_2 saa arvon 1.5434 ja alempi asetelmapiste x_1 saa arvon -1.5434. Näin ollen optimaalinen malli riittävän suurelle joukolle X on

$$\theta^* = \begin{bmatrix} -1.5435 & 1.5434 \\ 0.5 & 0.5 \end{bmatrix}, \quad (2.2.17)$$

jossa ensimmäisellä rivillä on optimaaliset asetelmapisteet eli D-optimaaliset tasot ($x_1 = -1.5435$ ja $x_2 = 1.5435$) ja toisella rivillä on niiden painot (kummankin paino on 0.5). Toisin sanoen optimaaliset asetelmapisteet eli tukipisteet (*support points*)

ovat samoin painotettuja sekä symmetrisiä pisteen $x = 0$ suhteen. Tukipisteisiin liittyvät keskiarvot ovat $\mu_1 = 0.176$ sekä $\mu_2 = 1 - 0.176 = 0.824$. Regressiomallin kaavasta (2.2.5) saadaan mallille ylempi tukipiste seuraavalla kaavalla, kun annetaan prediktorille η arvo x_0^*:

$$x_0^* = \frac{1.5434 - \beta_0}{\beta_1}. \tag{2.2.18}$$

Kun β_1 lähestyy nollaa, parametrin x_0^* arvo kasvaa rajatta. Tämä tarkoittaa sitä, että parametrin β_1 pienentyessä asetelmapisteet alkavat lähestyä arvoja $\pm\infty$. Käytännössä joukon X arvot eivät ole rajoittamattomia, jolloin asetelmapisteet lähestyvät joukon X ylä- ja alarajoja. Näin ollen optimaalinen malli arvolle $\beta_1 = 0$ asettaa tasaiset painot joukon X ala- ja ylärajoille. (Atkinson & Donev & Tobias, 2007.)

2.3 Suurimman uskottavuuden estimaattien olemassaoloehto

Binäärisen logistisen regressiomallin tarkoituksena on mallintaa binäärisen vasteen y_i $(i = 1, \dots, n)$ ja muuttujan x välistä yhteyttä. Tämä tapahtuu sovittamalla otospisteisiin n logistinen jakauma suurimman uskottavuuden menetelmällä. Jotta mallin parametrit pystyttäisiin estimoimaan suurimman uskottavuuden menetelmällä, täytyy suurimman uskottavuuden estimaattien olla olemassa. Tästä syystä on relevanttia tarkastella, milloin suurimman uskottavuuden estimaatit ovat olemassa. Tässä tutkielmassa keskitytään tapaukseen, jossa muuttuja x on yksiulotteinen. Tämä on moniulotteisen olemassaolon erikoistapaus. Moniulotteista tapausta ovat käsitelleet muun muassa Konis (2007) sekä Albert ja Anderson (1984).

Otospisteiden sanotaan olevan erillisiä eli separoituneita, kun on olemassa y-akselin suuntainen suora, joka erottelee toisistaan ne muuttujan x arvot, joille $y_i = 0$ ja joille $y_i = 1$. Toisin sanoen otospisteet, joille $y_i = 0$, jäävät suoran toiselle puolelle ja otospisteet, joille $y_i = 1$, jäävät suoran toiselle puolelle. Suoralle jääville otospisteille puolestaan pätee joko $y_i = 0$ tai $y_i = 1$. Moniulotteisessa tapauksessa voidaan määritellä hypertaso H, joka jakaa otospisteet erillisiin joukkoihin. Erillisyyden käsite liittyy suurimman uskottavuuden estimaattien olemassaolon määrittelemiseen olennaisesti, kuten myöhemmin tullaan huomaamaan.

Otospisteet voidaan jakaa kolmeen toisensa poissulkevaan kokoonpanoon, jotka ovat täysin erilliset otospisteet (*completely separated*), näennäisesti erilliset otospisteet *(quasicompletely separated*) ja päällekkäiset otospisteet *(overlapped*). Termi erillinen eli separoitunut, tarkoittaa otospisteitä, jotka ovat joko täysin tai näennäisesti erillisiä. Separoituneille otospisteille ei voida laskea yksikäsitteisiä

suurimman uskottavuuden estimaatteja, kun taas päällekkäisille otospisteille löytyvät yksikäsitteiset suurimman uskottavuuden estimaatit. Tarkastellaan seuraavaksi tarkemmin, mitä erillisyys ja päällekkäisyys täsmällisemmin tarkoittavat.

2.3.1 Täysi erillisyys

Otospisteet n ovat täysin erillisiä, jos seuraavat ehdot täyttyvät

i) $y_i = 0, \text{kun } x_i < c$

ii) $y_i = 1, \text{kun } x_i > c$

iii) joko $y_i = 0 \; \forall \; i, \text{joille } x_i = c$ tai $y_i = 1 \; \forall \; i, \text{joille } x_i = c$

missä c on jokin vakio välillä $[x_{\min}, x_{\max}]$ ja $i = 1, \dots, n$.

Lause 1. Jos otospisteet ovat täysin erillisiä, suurimman uskottavuuden estimaattia $\hat{\beta}$ ei ole olemassa (Albert & Anderson, 1984).

Otospisteet ovat siis täysin erillisiä silloin, kun vakion c kautta kulkeva y-akselin suuntainen suora, nimetään se suoraksi l, jakaa otospisteet kahdeksi täysin erilliseksi joukoksi vasteen arvojen perusteella. Binäärisen vasteen tapauksessa voidaan siis ajatella, että suoran l toiselle puolelle jäävät ne otospisteet, joiden vaste saa arvon 1. Toiselle puolelle tätä suoraa puolestaan jäävät ne otospisteet, joiden vasteen arvo on 0. Suora l siis erottaa otosjoukot toisistaan siten, että itse suoralla l, vaste voi saada vain jompaakumpaa arvoa, mutta ei molempia. Seuraavaksi on pari esimerkkiä täydestä erillisyydestä.

Esim.1. Vaste y saa arvon 0 kaikissa pisteissään, kun $x < 5$ ja vaste y saa arvon 1 kaikissa pisteissään, kun $x > 5$. Tässä tapauksessa, kun $x = 5$, niin vaste saa joko arvoja 0 tai arvoja 1, mutta ei molempia. Joukot ovat täysin erillisiä, koska ne voidaan erottaa y-akselin suuntaisella suoralla l, eikä suoralle osu sellaisia mittauksia, joissa vaste saisi sekä arvoja 0 että 1.

Esim.2. Vaste y saa vain arvoja 0 muuttujan x välin alemmassa päätepisteessä $x_{\min}$ ja vain arvoja 1 kaikissa muissa muuttujan x pisteissä välillä $(x_{\min}, x_{\max}]$. Tällöin suora l voidaan piirtää jälleen siten, että siihen osuu vain joko vasteen arvoja 1 tai 0. Vastaavasti tapahtuu silloin, kun vaste y saa vain arvoja 1 muuttujan x välin ylemmässä päätepisteessä $x_{\max}$ ja vain arvoja 0 kaikissa muissa muuttujan x pisteissä välillä
$[x_{\min}, x_{\max})$.

2.3.2 Näennäinen erillisyys

Jos ei ole olemassa vakion c kautta kulkevaa y-akselin suuntaista suoraa l, joka erottelee täysin otospisteet n, saattaa kyseessä olla näennäinen erillisyys. Otospisteet ovat näennäisesti erillisiä, jos seuraavat ehdot pätevät

i) $y_i = 0, \text{kun } x_i < c$

ii) $y_i = 1, \text{kun } x_i > c$

iii) $\exists\, i, j, \text{joille } x_i = x_j = c \text{ ja } y_i = 0 \text{ ja } y_j = 1$

Lause 2. Jos otospisteet ovat näennäisesti erillisiä, silloin suurimman uskottavuuden estimaattia $\hat{\beta}$ ei ole olemassa (Albert & Anderson, 1984).

Otospisteet ovat siis näennäisesti erillisiä silloin, kun suora l jakaa otospisteet kahdeksi erilliseksi joukoksi vasteen arvojen perusteella niin, että otospisteet ovat päällekkäisiä vain itse suoralla. Moniulotteisessa tapauksessa vastaavasti hypertaso H erottelee otospisteet eri joukoiksi ja päällekkäisyyttä on vain hypertasolla. Yksiulotteisessa tapauksessa ajatellaan jälleen, että on olemassa y-akselin suuntainen suora, joka jakaa otospisteet kahdeksi joukoksi. Toisella puolella ovat edelleen ne pisteet, joiden vasteen arvo on 1 ja toisella puolella ne pisteet, joiden vasteen arvo on 0. Pisteessä c, jossa suora leikkaa x-akselin, otospiste voi kuitenkin saada sekä arvon 1 että arvon 0. Seuraavaksi on pari esimerkkiä näennäisestä erillisyydestä.

Esim.3. Vaste y saa vain arvoja 0, kun $x \leq 5$ ja vaste saa vain arvoja 1, kun $x \geq 5$. Tällöin pisteessä $x = c = 5$ vaste voi saada sekä arvon 0 että arvon 1.

Esim.4. Vaste y saa arvoja 0 ja 1 muuttujan x välin alemmassa päätepisteessä $x_{\min}$ ja vain arvoja 1 kaikissa muissa muuttujan x pisteissä välillä $(x_{\min}, x_{\max}]$. Tällöin suora l voidaan piirtää arvolla $x_{\min}$ siten, että suoran päälle osuu sekä vasteen arvoja 0 että 1. Vastaavasti tapahtuu silloin, kun vaste y saa vain arvoja 0 ja 1 muuttujan x välin ylemmässä päätepisteessä $x_{\max}$ ja vain arvoja 0 kaikissa muissa muuttujan x pisteissä välillä $[x_{\min}, x_{\max})$.

2.3.3 Päällekkäisyys

Mikäli otospisteet eivät ole täysin erillisiä tai näennäisesti erillisiä, niiden sanotaan olevan päällekkäisiä.

Lause 3. Kun otospisteet ovat päällekkäisiä, on olemassa suurimman uskottavuuden estimaatti $\hat{\beta}$, joka on yksikäsitteinen (Silvapulle, 1981).

Otospisteet ovat siis päällekkäisiä silloin, kun ei voida määritellä y-akselin suuntaista suoraa l, joka erottelisi otospisteet joko täysin erillisiksi tai näennäisesti erillisiksi joukoiksi. Vastaavasti moniulotteisessa tapauksessa ei ole hypertasoa H, joka jakaisi otospisteet vasteen arvojen perusteella joko täysin erillisiksi joukoiksi tai näennäisesti erillisiksi joukoiksi. Tämä tarkoittaa sitä, että vaste saa arvoja 1 ja 0 kahdessa tai useammassa pisteessä. Seuraavaksi on pari esimerkkiä päällekkäisyydestä.

Esim.5. Ajatellaan tapausta, jossa vaste y saa vain arvoja 0, kun $x \leq 3$ ja vaste saa vain arvoja 1, kun $x \geq 6$. Tällöin välillä [4,5], vaste saa sekä arvoja 0 että 1. Koska vaste saa arvoja 0 ja 1 useammassa kuin yhdessä pisteessä, ei voida piirtää suoraa l, joka jakaisi pisteet niin, että suoran molemmin puolin olisi vain yhtä arvoa ja suoralla joko yhtä tai kumpaakin arvoa.

Esim.6. Ajatellaan seuraavaksi tapausta, jossa vaste y saa vain arvoja 0, kun $x \leq 2$ ja $3 < x \leq 4$, sekä arvoja 1, kun $2 < x \leq 3$ ja $x > 4$. Tällöin pystytään kyllä piirtämään suora l arvolla $x = 3$, jossa vaste saa vain arvoja 1, mutta tämä suora ei erottele otospisteitä kahdeksi erilliseksi joukoksi vasteen arvojen perusteella. Tässä tapauksessa on myös kyse päällekkäisyydestä.

2.4 Binäärinen etsintäalgoritmi

Binäärinen etsintäalgoritmi esiteltiin ensimmäisen kerran artikkelissa Karvanen ym. (2007), jonka jälkeen sitä tarkasteltiin tarkemmin artikkelissa Karvanen (2008). Tämä luku pohjautuu näihin kahteen artikkeliin. Binäärisen etsintäalgoritmin ideana on mitata mahdollisimman nopeasti sellaiset selittäjän arvot, joille otospisteet ovat päällekkäisiä. Tällä tavoin suurimman uskottavuuden estimaatit voidaan löytää mahdollisimman nopeasti. Kun suurimman uskottavuuden estimaatit on löydetty, koetta voidaan jatkaa käyttämällä mallia, joka on mitattujen otospisteiden perusteella optimaalinen.

Jotta suurimman uskottavuuden estimaatit olisivat olemassa, täytyy otospisteiden olla päällekkäisiä. Tätä varten pitää löytää kaksi selittäjän x arvoa, joissa vaste saa arvoja 0 ja 1. Binäärinen etsintäalgoritmi etsii ensin yhtä sellaista selittäjän arvoa, jossa vasteen arvoksi voidaan mitata sekä arvo yksi että nolla. Jokaisella askeleella vasteen arvo mitataan sen hetkisen välin keskikohdasta. Jos vasteen arvoksi saadaan pelkkiä nollia, keskikohta otetaan uudeksi välin alkupisteeksi. Jos puolestaan vasteen arvoksi mitataan vain lukuja yksi, keskikohta otetaan välin uudeksi päätepisteeksi. Tämä ensimmäinen osa lopettaa etsintänsä, kun vasteen arvoksi on saatu samassa pisteessä x sekä lukuja yksi että nolla. Toinen selittäjän arvo, joka takaa otospisteiden päällekkäisyyden, löytyy yleensä selittäjän x lähiympäristöstä. Algoritmi voidaan kirjoittaa seuraavasti:

1. Käytetään aiempaa tietoa muuttujasta x välin $[x_{min}, x_{max}]$ määrittämiseksi, tällöin voidaan olla varmoja että $F(x_{max}; \beta_0, \beta_1) - F(x_{min}; \beta_0, \beta_1)$ on lähellä lukua 1. Toisin sanoen väli valitaan siten, että se sisältää kaikki mielekkäät selittäjän x arvot. Yleensä pystytään määrittämään edes jonkinlainen väli, mutta mikäli aiempi tieto muuttujasta x on epätarkkaa, saattaa tämä väli olla hyvinkin suuri.

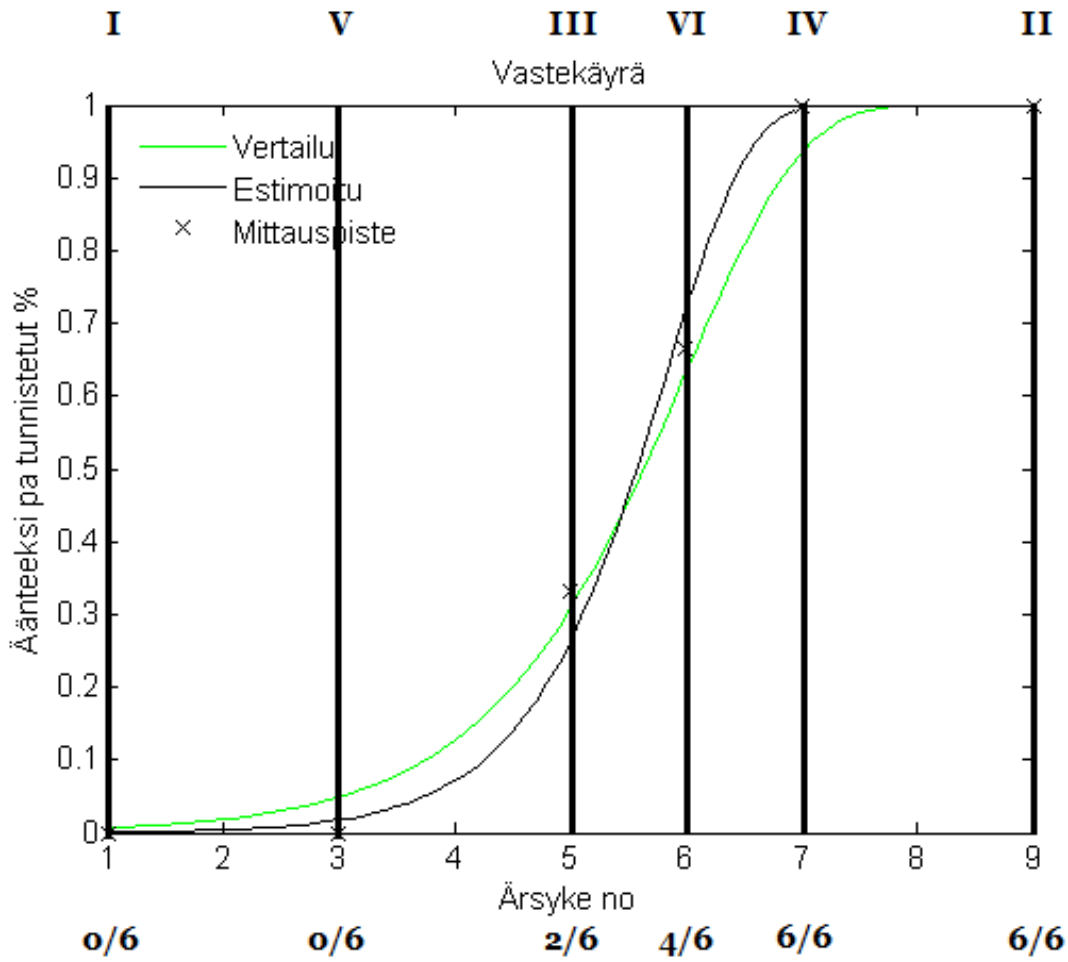
2. Etsitään binäärisellä haulla piste x väliltä $[x_{min}, x_{max}]$, jossa vasteen arvoiksi mitataan lukuja yksi ja nolla. Jotta tämä olisi mahdollista, vasteen arvo täytyy mitata vähintään kahdesti jokaisessa pisteessä. Ensimmäisellä kerralla vasteen arvot mitataan välin $[x_{min}, x_{max}]$ päätepisteissä. Tästä eteenpäin vasteen arvo mitataan sen hetkisen välin $[x_l, x_u]$ keskikohdasta eli $x = (x_l + x_u)/2$. Jos vasteen arvoksi välin keskikohdassa saadaan pelkkiä nollia, keskikohta otetaan uudeksi välin alkupisteeksi. Jos puolestaan vasteen arvoksi mitataan vain lukuja yksi, keskikohta otetaan välin uudeksi päätepisteeksi.

3. Määritellään $|\varepsilon| = (x_u - x_l)/4$. Etumerkki muuttujalle ε määräytyy mitatun vasteen mukaan: $sign(\varepsilon) = sign(0.5 - \bar{y})$, missä $\bar{y}$ on vasteiden keskiarvo pisteessä x. Jos $0.5 - \bar{y} = 0$ eli vasteen arvoja nolla ja yksi on mitattu saman verran, muuttujan ε etumerkki valitaan satunnaisesti.

4. Mitataan vasteet ensin pisteessä $x + \varepsilon$. Jos vasteen arvoiksi ei mitata sekä lukuja nolla että yksi, mitataan vasteet myös pisteessä $x - \varepsilon$.

5. Jos on löytynyt kaksi selittäjän x arvoa, joissa vaste saa arvoja nolla ja yksi, niin edetään suurimman uskottavuuden estimointiin. Muussa tapauksessa jaetaan ε kahdella ja palataan askeleeseen 4.

Kuvassa 2 on havainnollistettu algoritmin toimintaa. Selittäjänä x on 9 eri ääntä siten, että ääni 1 vastaa äännettä /ba/ ja ääni 9 vastaa äännettä /pa/. Välissä olevat arvot vastaavat ääniä, jotka ovat näiden kahden äänteen välimuotoja. Esimerkiksi ääni numero 5 on puoliksi äänne /ba/ ja puoliksi äänne /pa/. Tähän perehdytään tarkemmin seuraavassa kappaleessa. Kuvassa on vasteen arvona y prosenttimääränä ne kerrat, kuinka monesti tutkittava on vastannut kuulemansa äänen olevan /pa/. Vertailukuvaaja on ennalta määrätty logistinen jakauma, joka kuvaa sitä, miten vastekäyrän odotetaan käyttäytyvän. Estimoitu kuvaaja puolestaan kuvaa saatuja havaintoja. Pystysuorat viivat kuvassa esittävät mitattuja



Kuva 2: Esimerkki binäärisestä etsintäalgoritmista, johon on sovitettu vastekäyrä. Pystysuorat viivat esittävät mitattuja selittäjätasoja ja roomalaiset numerot kertovat mittausten järjestyksen. Viivan alapuolella oleva luku kertoo, montako kuudesta mitatusta vasteesta sai arvon 1. Suurimman uskottavuuden estimointiin tarvittavat kaksi pistettä löytyivät mittauskerroilla III ja VI.

selittäjätasoja ja roomalaiset numerot kertovat mittausten järjestyksen. Viivan alapuolella oleva luku kertoo, montako kuudesta mitatusta vasteesta sai arvon 1 (äänne /pa/).

Ensimmäisenä kuvan 2 tapauksessa tutkitaan välin päätepisteet eli 1 ja 9. Koska kummassakin pisteessä on mitattu vain yhtä vasteen arvoa, seuraavana tutkitaan puoliväli eli piste 5. Koska tässä pisteessä vaste saa arvoja 0 ja 1, voidaan siirtyä seuraavaan askeleeseen eli tarkastellaan neljäsosan päässä olevia selittäjän arvoja. Neljäsosa välin pituudesta on 2 ($\varepsilon = 2$) eli seuraava mittauspiste on joko 3 tai 7. Koska $0.5 - (2/6) > 0$, niin luvun ε merkki on positiivinen ja seuraava tarkastelupiste on 7. Tässä pisteessä vaste on saanut vain arvoja 1, joten seuraavana tutkitaan piste 3. Koska myöskään tässä pisteessä vaste ei ole saanut muuta arvoa kuin 0, jaetaan luku ε kahdella. Seuraavaksi tarkastelupisteeksi saadaan näin ollen 6, jossa neljällä kerralla kuudesta vaste on saanut arvon 1. Suurimman uskottavuuden estimointiin tarvittavat kaksi pistettä on näin ollen löydetty ja etsintäalgoritmi saadaan päätökseen.

Alun perin binäärinen etsintäalgoritmi on suunniteltu jatkuvalle selittäjälle x. Tässä tutkielmassa x on kuitenkin diskreetti ja saa vain kokonaislukuarvoja. Ongelmana tässä tapauksessa on, että ääniä pitää olla riittävän paljon, jotta algoritmi toimisi mielekkäästi. Tässä tutkielmassa ääniä on 9 kappaletta, joka alkaa olla lähellä minimiä äänten määrälle. Esimerkiksi alle 5 äänen tapauksessa algoritmia ei ole enää järkeä käyttää. Hyvä määrä voisi olla lähemmäs 20 ääntä, mutta luonnollisesti mitä enemmän ääniä, sitä järkevämpää algoritmin käyttö on. Toinen ongelma diskreetin selittäjän kanssa on, että etsintäalgoritmi saattaa löytää arvoja, joita ei ole määritelty. Esimerkiksi, jos on 8 kokonaislukuarvoa välillä 1-8, niin keskimmäinen luku on 4.5, joka ei ole kokonaisluku. Tällöin täytyy päättää, mitä tehdään tällaisille etsintäalgoritmin antamille lukuarvoille, jotka osuvat kahden luvun väliin. Luonnollisin vaihtoehto kokonaislukujen tapauksessa on pyöristää arvo lähimpään kokonaislukuun.

Kun suurimman uskottavuuden estimaatit on löydetty algoritmin avulla, voidaan ne laskea aineistosta kahdella tavalla. Estimaatit voidaan laskea koko aineistosta, jonka binäärinen etsintäalgoritmi on kerännyt tai voidaan käyttää välin päätepisteitä ja kahta pistettä, jotka löydettiin algoritmin askeleilla 2 ja 4. Tässä tutkielmassa suurimman uskottavuuden estimaatit parametreille β_0 ja β_1 lasketaan aina koko siitä aineistosta, joka on ehditty estimointihetkeen mennessä kerätä. Tällöin, kun suurimman uskottavuuden estimaatit ovat löytyneet, niiden arvo päivitetään jokaisen toiston jälkeen kaikesta siihen asti kerätystä aineistosta. Erityisen kiinnostavat estimaattien arvot ovat tässä tapauksessa alustavat estimaatit ja lopulliset estimaatit. Alustavilla estimaateilla tarkoitetaan niitä estimaatteja, jotka löytyvät etsintäalgoritmin avulla ja lopulliset estimaatit ovat ne, jotka lasketaan lopuksi koko aineistosta.

Kuten artikkelissa Karvanen ym. (2007), jakauman parametrien β_0 ja β_1 lisäksi lasketaan parametrit θ ja λ, jotka kuvaavat vastekäyrän sijaintia ja leveyttä. Keskikohta θ saadaan yhtälöstä

$$\theta = \frac{1}{\beta_0}(F(0.5) - \beta_1) \tag{2.4.1}$$

ja käyrän leveys λ saadaan yhtälöstä

$$\lambda = \frac{1}{\beta_0}\big(F(0.9) - F(0.1)\big). \tag{2.4.2}$$

3 Kategorisen havaitsemisen tutkiminen

Tässä luvussa esitellään binäärisen etsintäalgoritmin soveltamista kategorisen havaitsemisen tutkimiseen. Aluksi luvussa 3.1 esitellään, mitä on äänteiden kategorinen havaitseminen ja sen jälkeen luvussa 3.2 esitellään tutkimuksen kulkua käytännössä.

3.1 Äänteiden kategorinen havaitseminen

Luku pohjautuu kirjaan Harley (2001). Puhekieli koostuu foneemeista eli äänteistä, jotka ovat äänen perusyksiköitä. Foneemit ovat kielikohtaisia, jolloin eri kielet koostuvat erilaisista foneemeista. Koska foneemit muodostavat kielen, yhden äänteen muuttaminen toiseksi muuttaa sanan merkitystä. Äänteiden akustiset ominaisuudet (miltä äänne kuulostaa) eivät ole kiinteitä, vaan ne saattavat vaihdella esimerkiksi kontekstin tai puheen nopeuden mukaan. Vaikka äänteet saattavat kuulostaa erilaisilta, ihminen harvoin huomaa näitä eroavaisuuksia. On havaittu, että ihminen on taipuvainen luokittelemaan puheäänet joko yhdeksi tai toiseksi foneemiksi, eikä näiden kahden äänteen välimuodoksi. Tätä ilmiötä kutsutaan äänteiden kategoriseksi havaitsemiseksi. Liberman ym. (1957) todistivat ilmiön ensimmäisenä luomalla puhesyntetisaattorilla jatkumon keinotekoisia tavuja, jotka erosivat toisistaan ääntämispaikan suhteen. Jatkumosta huolimatta koehenkilöt sijoittivat nämä tavut kolmeen selvästi toisistaan eroavaan kategoriaan.

Toinen esimerkki kategorisesta havaitsemisesta on soinnin alkamisaika (voice onset time, VOT). Soinnillisissa konsonanteissa (esimerkiksi b ja d) äänihuulet alkavat värähdellä heti, kun ääntöväylä on sulkeutunut. Soinnittomissa konsonanteissa (esimerkiksi p ja t) värähtelyn alkamisessa on noin 60 millisekunnin viive. Tätä viivettä on kuitenkin mahdollista muokata, jolloin saadaan jälleen aikaiseksi jatkumo. Mikäli soinnin alkamisajaksi asetetaan 30 millisekuntia, äänne luokitellaan kuitenkin joko soinnilliseksi tai soinnittomaksi. Kumpaan kategoriaan äänne luokitellaan, vaihtelee eri aikoina ja eri ihmisillä.

Äänteiden kategorisessa havaitsemisessa on olemassa jokaiselle ihmiselle yksilöllinen kynnysarvo, jossa henkilö ei pysty erottamaan kahta äännettä toisistaan. Tämä kynnysarvo voidaan havaita siten, että toistettaessa samaa ääntä useamman kerran, kynnysarvon kohdalla koehenkilö luokittelee osan saman äänen toistoista toiseen kategoriaan ja osan taas toiseen kategoriaan.

3.2 Koeasetelma

Tarkoituksena on tutkia äänteiden kategorista havaitsemista äänneparin /ba/ ja /pa/ avulla siten, että koehenkilöille soitetaan näitä kahta äännettä sekä näiden välimuotoja. Ääniä on tässä tapauksessa kokeessa yhteensä 9 kappaletta ja ne ovat Tiina Parviaisen tekemät. Koska äänneparina on /ba/-/pa/, niin jatkumo äänteiden välille muodostetaan muokkaamalla soinnin alkamisaikaa (VOT). Äänet on tehty nauhoittamalla sanelukoneella englantia äidinkielenään puhuvalta mieheltä äänteet /ba/ ja /pa/. Tämän jälkeen äänet on käsitelty Adobe Audition ohjelmalla siten, että /pa/-äänen alusta katkaistiin akustista signaalia aina pieni pätkä (asteittain kasvattaen pätkän pituutta) ja tämä pätkä lisättiin /ba/-äänen alkuun. Tällöin saatiin jatkumo ääniä, joissa ensimmäisessä soinnin alkamisaika on 0 ms (äänne /ba/) ja viimeisessä soinnin alkamisaika on 64 ms (äänne /pa/). Näiden välissä olevissa äänissä lisättyjen pätkien pituudet (soinnin alkamisajat) ovat 5, 9, 17, 25, 34, 44 ja 57 millisekuntia.

Ajatellaan johdannossa ollutta kuulokynnysesimerkkiä, mutta vaihdetaan x-akselille desibelien tilalle äänet 1-9 (1=/ba/ ja 9=/pa/) ja y-akselille keskiarvo vasteista *y* jokaisessa pisteessä *x*. Vaste *y* saa tässä tapauksessa arvon 0, kun kuultu ääni tulkitaan äänteeksi /ba/ ja arvon 1, kun ääni tulkitaan äänteeksi /pa/. Nyt, jos piirretään kuvaaja koehenkilön havaitsemisesta, niin se noudattaa logistista jakaumaa, samalla tavalla kuin johdannon kuulokynnysesimerkissä. Oletus siis on, että ensin kaikki äänet on luokiteltu kategoriaan 0, sitten osa äänistä on luokiteltu kategoriaan 0 ja osa kategoriaan 1 ja lopuksi kaikki äänet on luokiteltu kategoriaan 1.

Yleinen lähestymistapa äänteiden kategorisen havaitsemisen tutkimiseen on ollut, että kaikille koehenkilöille määritellään kolme eri kategoriaa: ensimmäisessä kategoriassa ovat äänteet, jotka luokitellaan äänteeksi /ba/. Toisessa kategoriassa ovat ne äänteet, jotka luokitellaan kuuluvaksi kumpaankin äännekategoriaan. Kolmannessa kategoriassa puolestaan ovat äänteet, jotka luokitellaan äänteeksi /pa/. Yleensä nämä kolme kategoriaa ovat yhtä suuret ja samat kaikille koehenkilöille. Esimerkiksi tässä tapauksessa kategoriat olisivat 1-3, 4-6 ja 7-9. Koska äänteiden kategorinen havaitseminen kuitenkin on jokaisella ihmisellä yksilöllistä, niin tässä tutkielmassa on haluttu muodostaa kategoriat jokaisen koehenkilön kohdalla erikseen, kaikille yhteisten ja kiinnitettyjen kategorioiden sijasta. Tässä kategorioiden muodostamisessa on käytetty hyväksi binääristä etsintäalgoritmia.

3.2.1 Koe- ja kontrollikäsittelyt

Koekäsittelyssä muokattuja ääniä soitetaan koehenkilöille ja heidän pitää jokaisen äänen kohdalla kertoa hiiren painikkeita painamalla, onko kyseessä äänne /ba/ vai /pa/. Kokeen aikana koehenkilöiltä mitataan aivojen toimintaa EEG:llä. Soitettujen

äänien järjestys jokaisen koehenkilön kohdalla tallennetaan, jotta äänet voidaan soittaa uudelleen samassa järjestyksessä kontrollikäsittelyn yhteydessä. Äänten sama järjestys molemmissa käsittelyissä johtuu siitä, että kummankin käsittelyn vaikutuksia suhteessa toisiinsa halutaan verrata keskenään, jolloin äänten pitää olla samassa järjestyksessä.

Kontrollikäsittelyssä koehenkilölle soitetaan samat äänet samassa järjestyksessä kuin koekäsittelykerralla, mutta tällä kertaa hänellä ei ole tehtävää, että äänet pitäisi luokitella äänteeksi /ba/ tai /pa/. Sen sijaan kontrollikäsittelyn aikana koehenkilön täytyy vain passiivisesti kuunnella ääniä ja samalla häneltä mitataan edelleen aivojen toimintaa EEG:llä.

3.2.2 Kritiikkiä koeasetelmaa koskien

Koeasetelmaa kohtaan voidaan esittää kritiikkiä kolmen asian suhteen. Ensimmäisenä on toistovaikutus, joka johtuu siitä, että äänet soitetaan kummassakin käsittelyssä samassa järjestyksessä. Koska ääniä on kuitenkin useammanlaisia ja EEG:n tarkastelussa keskitytään myöhäisiin vasteisiin (yli 100 millisekuntia havainnon jälkeen), niin toistovaikutuksella ei pitäisi olla vaikutusta tuloksiin.

Toiseksi tiedostetaan, että soittamalla kaikille koehenkilöille käsittelyt samassa järjestyksessä, saattaa tuloksissa näkyä oppimisvaikutusta, eikä käsittelyiden järjestyksen vaikutusta tuloksiin voida näin ollen poissulkea. Koe- ja kontrollikäsittelyiden tekeminen toisin päin ei kuitenkaan tässä koeasetelmassa ole mahdollista, koska äänten järjestys on riippuvainen vasteista.

Kolmas ongelma on, että äänet on alun perin tarkoitettu englanninkielisille tehtävään tutkimukseen. Tästä seuraa se, että suomalaisen korvaan kuulostaa siltä, kuin kategorioita olisi kolme: /ba/, /pa/ ja /pha/. Alkupään äänet kuulostavat äänteiltä /ba/ sekä /pa/ ja loppupään äänet kuulostavat äänteeltä /pha/. Tästä voi seurata ongelmia äänteiden tunnistamiseen.

3.2.3 Tavoitteet

Tutkimuksella on kaksi erilaista tavoitetta: toinen on neuropsykologiaan ja kategoriseen havaitsemiseen liittyvä tavoite ja toinen on koesuunnitteluun liittyvä tavoite. Neuropsykologian kannalta kokeen tavoitteena on vertailla, mitä aivojen toiminnassa tapahtuu koe- ja kontrollikäsittelyjen aikana. Halutaan tietää, onko kategorinen havaitseminen ilmiönä olemassa myös silloin, kun ihmisellä ei ole tehtävänä luokitella ääniä eri äänneluokkiin. Halutaan siis tutkia, johtuuko kategorinen havaitseminen ilmiönä tehtävästä vai tapahtuuko aivoissa tällaista kategorista luokittelua muutenkin.

Koesuunnittelun kannalta kokeen tarkoituksena on pyrkiä kehittämään sellainen menetelmä, että itse koetilanne saadaan optimoitua. Tämä tarkoittaa sitä,

että halutaan kehittää vasteisiin perustuva järjestelmä, jolla tarvittavien toistojen määrä voitaisiin optimoida erikseen jokaisen koehenkilön kohdalla. Monesti neuropsykologisissa tutkimuksissa ääniä saatetaan toistaa tuhansia kertoja, jolloin kokeiden kestokin saattaa olla monta tuntia. Kuitenkin rahallisesti ja ajankäytön kannalta paras ratkaisu olisi, että toistojen määrä saataisiin sidottua tarvittavien tulosten määrään. Erityisesti silloin, kun tutkitaan sellaisia ilmiöitä, joissa vaste on yksilöllinen, kuten kategorinen havaitseminen tässä tapauksessa. Ajatellaan esimerkiksi tapausta, jossa koehenkilölle A tarvitaan 500 toistoa, jotta saadaan riittävästi aineistoa tulosten analysointiin ja henkilöille B ja C tarvittavien toistojen lukumäärä on 1000 ja 650. Tavallisin lähestymistapa tässä tilanteessa on soittaa kaikille varmasti riittävä määrä ääniä, esimerkiksi 1500 toistoa jokaiselle koehenkilölle. Tästä kuitenkin nähdään, että optimaalisessa tilanteessa koehenkilölle A olisi riittänyt 1000 toistoa vähemmän, koehenkilölle B olisi riittänyt 500 toistoa vähemmän ja koehenkilölle C olisi riittänyt 850 toistoa vähemmän. Tästä näkee, että koetilanne saattaa olla hyvinkin kaukana optimaalisesta, jolloin kokeen kustannukset nousevat turhaan.

Tässä tutkielmassa koetilanteessa yritetään optimoida kiinnostavien äänten toistojen määrää. Kiinnostavien äänten toistojen määrää on optimoitu pyrkimällä tunnistamaan nämä äänet mahdollisimman nopeasti ja toistamalla niitä kaksinkertainen määrä suhteessa muihin ääniin. Lisäksi tässä tutkielmassa kiinnostavien äänten määrä on kiinnitetty eli niitä määritellään jokaiselle koehenkilölle 3. Poikkeuksena on tilanne, jossa kiinnostavia ääniä ei pystytä määrittämään. Tällöin kaikkia ääniä pidetään yhtä kiinnostavina ja niitä toistetaan kaikkia saman verran.

4 Ohjelma kategorisen havaitsemisen tutkimiseen

Tässä luvussa perehdytään tarkemmin MATLAB:lla tehtyyn ohjelmaan, jolla tutkitaan kategorista havaitsemista. Tässä esitellään vain ne koodit, jotka olen itse tehnyt tai joita olen muokannut, kokonaisuudessaan ohjelmakoodi löytyy liitteistä. Ohjelma on toteutettu englanninkielisenä. Ohjelman pohjana on käytetty Juha Karvasen tekemää optdesign-ohjelmaa, jossa on koodi binääriselle etsintäalgoritmille sekä suurimman uskottavuuden estimaattien laskemiselle, kun jakaumana on Gombertz-jakauma. Binääristä etsintäalgoritmia on muokattu, koska se ei sellaisenaan soveltunut kategorisen havaitsemisen tutkimiseen. Näitä muutoksia käsitellään tarkemmin tässä luvussa. Luvussa 4.1 tarkastellaan pääohjelman toimintaa ja luvussa 4.2 käydään läpi aliohjelmat ja niiden toiminta.

4.1 Pääohjelma – perceptionTest

Pääohjelmalle on annettu nimeksi perceptionTest ja se on suunniteltu nimenomaan äänteiden kategorisen havaitsemisen tutkimiseen. Ohjelma toistaa ääniä koehenkilölle, jotka koehenkilön pitää luokitella kahteen kategoriaan, joko äänteeksi /ba/ tai /pa/. Ohjelma rekisteröi koehenkilön vastaukset, nämä ovat vasteen *Y* arvot. Vaste on siis binäärinen eli saa arvoja nolla ja yksi, kuten tässä tutkielmassa on aiemmin esitetty. Arvo nolla vastaa äännettä /ba/ ja arvo yksi äännettä /pa/.

Kun ohjelma käynnistyy, se kysyy ensin halutaanko koe suorittaa, jos vastaus on ”kyllä” (yes), ohjelma kysyy tallennettavalle tiedostolle annettavaa nimeä. Mikäli vastaus on ”ei” (no), ohjelma sulkeutuu. Pääohjelma alustaa aivan ensimmäisenä joitakin muuttujia, joista käyttäjän kannalta tärkeitä ovat *minlimit* ja *maxlimit,* jotka määrittävät ylä- ja alarajan prediktorille *x*. Tässä tutkielmassa nämä ovat 1 ja 9, jolloin ääniä on näin ollen yhdeksän erilaista. Numero 1 on äänne /ba/ ja numero 9 on äänne /pa/.

Pääohjelma voidaan jakaa kahteen osaan: koeosaan ja kontrolliosaan. Koeosa tulee aina ennen kontrolliosaa ja koeosan jälkeen ohjelma on mahdollista lopettaa. Koeosassa alustetaan ensimmäisenä tarvittavat parametrit valmiiksi. Käyttäjän kannalta merkittäviä parametreja ovat *nstage*, *xplot, repeat* ja *n_inc*. Parametri *nstage* määrittelee, montako kertaa binääristä etsintäalgoritmia enimmillään kutsutaan ennen kuin ohjelman suoritus siirtyy eteenpäin. Parametriin *xplot* puolestaan annetaan tiedot (alaraja, skaala, yläraja) kuvaajan x-akselin piirtämistä varten. Parametri *repeat* kertoo, montako toistoa kaikkia ääniä vähintään soitetaan, lisäksi se kertoo, montako toistoa lisätään kiinnostaviin ääniin. Tässä tutkielmassa yhtä ääntä halutaan toistaa vähintään 42 kertaa jokaista. Tämän lisäksi kiinnostavimpia kolmea ääntä halutaan toistaa jokaista vielä 42 kertaa lisää, eli niitä

toistetaan jokaista 84 kertaa. Mikäli kiinnostavia ääniä ei pystytä määrittämään, jokaista yhdeksästä äänestä toistetaan 56 kertaa. Kokonaisuudessaan toistoja tulee tässä koeasetelmassa siis 504 kappaletta kummassakin tapauksessa. Parametriin *n_inc* puolestaan tulee tieto siitä, montako kertaa binäärinen etsintäalgoritmi toistaa yhtä ääntä yhden estimointikerran aikana.

Koeosan käynnistyessä luodaan valmiiksi kaksi ikkunaa. Näistä toinen on vastaamiselle ja siinä on kahdeksankulmio, joka vaihtaa väriä sen mukaan, saako koehenkilö painaa vastausnappulaa vai ei. Kahdeksankulmio on punainen, kun vastausta ei huomioida ja vihreä, kun vastaus huomioidaan. Toinen ikkuna on kuvalle, johon piirretään kuvaaja binäärisen etsintäalgoritmin saamista vastauksista ja kuvaaja kaikista kokeen aikana saaduista vastauksista. Kuvaajat yritetään piirtää jokaisen binäärisen etsintäalgoritmin kierroksen jälkeen, sekä kiinnostavien pisteiden valitsemisen jälkeen jokaisen toiston yhteydessä. Mikäli kuvan piirtäminen ei jostain syystä onnistu, niin se voidaan generoida myös jälkikäteen liitteistä löytyvällä ohjelmalla *draw*.

Seuraavaksi koeosassa suoritetaan mallin sovittaminen binäärisen etsintäalgoritmin avulla. Etsintäalgoritmi on jaettu kahteen osaan. Ensimmäisessä osassa halutaan tarkastella prediktorin *x* ylä- ja alarajaa sekä niiden läheisyydessä sijaitsevia prediktoritasoja. Tämä on erilaisuutensa vuoksi toteutettava erikseen. Tässä tutkielmassa ensimmäisessä osassa etsintäalgoritmia toistetaan ensin alarajaa 1 sekä sen läheltä ääniä 3 ja 5, joista jokaisesta tulee 3 toistoa. Tämän jälkeen toistetaan ylärajaa 9 sekä sen läheltä ääniä 5 ja 7, jokaista näistä tulee myös kolme toistoa.

Seuraavaksi siirrytään binäärisen etsintäalgoritmin toiseen osaan, joka vastaa luvussa 2.4 esiteltyä algoritmia, mikäli ensimmäisen laskentakerran aikana ei jo löytynyt vähintään kahta kiinnostavaa pistettä. Lisäksi huomattava on, että oletusten voimassaolon takia etsintäalgoritmin toinen osa suoritetaan vain, jos alarajalla mitattujen vasteiden keskiarvo on pienempi kuin ylärajalla mitattujen vasteiden keskiarvo. Mikäli binäärinen etsintäalgoritmi ei onnistu löytämään suurimman uskottavuuden estimaatteja, siirtyy se ohjelman suorituksessa eteenpäin, kun se on yrittänyt estimointia muuttujan *nstage* määrittämän lukumäärän verran.

Kun binäärinen etsintäalgoritmi on lopettanut etsintänsä, siirrytään kiinnostavien pisteiden valitsemiseen. Nämä kiinnostavat pisteet tallennetaan muuttujaan *interest_x_points*. Kiinnostavien pisteiden valinta tapahtuu seuraavalla algoritmilla:

- Binäärisen etsintäalgoritmin tarjoamasta mallista ei saada yhtään pistettä, jolloin joko
    a) kaikkien arvojen ollessa pelkkää nollaa tai ykköstä, toistetaan kaikkia ääniä saman verran tai
    b) osan ollessa pelkkiä nollia ja osan ollessa pelkkiä ykkösiä, niin valitaan kiinnostaviksi pisteiksi suurin nollia sisältävä arvo (*max0*), pienin ykkösiä sisältävä arvo (*min1*) sekä näiden vierestä yksi piste keskelle päin.

- Jos mallista saadaan yksi piste, niin
  a) valitaan yksi piste kummaltakin puolelta tai
  b) päädyissä valkataan kaksi vierekkäistä pistettä keskikohdan puolelta.
- Jos mallista saadaan 2 pistettä, niin
  a) vierekkäisille pisteille valitaan kolmas piste keskikohdan puolelta,
  b) jos pisteiden välissä on yksi piste, valitaan tämä piste kolmanneksi kiinnostavaksi pisteeksi tai
  c) jos pisteiden välissä on enemmän kuin kaksi pistettä, niin valitaan lähempänä keskikohtaa oleva piste ja kaksi pistettä sen vierestä keskikohdan puolelta. Mikäli toinen piste on keskikohdassa, valitaan se ja yksi piste sen kummaltakin puolelta.
- Jos mallista saadaan 3 pistettä, niin
  a) vierekkäiset pisteet valitaan suoraan kiinnostaviksi pisteiksi tai
  b) muussa tapauksessa valitaan kolmesta pisteestä keskimmäinen ja yksi piste sen kummaltakin puolelta.
- Jos mallista saadaan 4 pistettä, niin otetaan kahdesta keskimmäisestä pisteestä keskiarvo, pyöristetään se ja valkataan näin saadun pisteen kummaltakin puolelta yksi piste.
- Jos mallista saadaan enemmän kuin 4 pistettä, niin kaikki pisteet tulkitaan kiinnostaviksi pisteiksi, jolloin kaikkia ääniä toistetaan satunnaisesti saman verran.

Kiinnostavien pisteiden valitsemisen jälkeen ohjelma asettaa oikean toistojen määrän jokaiselle äänelle, sekoittaa äänet ja soittaa ne koehenkilölle. Kun kaikki toistot on suoritettu, niin ohjelma tallentaa tulokset. Excel-tiedostoon tallennetaan neljä eri välilehteä: ”Experiment”, ”Control”, ”Statistics” ja ”Other”. ”Experiment”-välilehdellä on soitettujen äänten järjestys sekä niihin liittyvät vasteen arvot ja ”Control”-välilehdellä on soitettujen äänten järjestys (sama kuin kokeessa). ”Statistics”-välilehdellä puolestaan on tietoja mallista, kuten estimoidut mallin parametrit. Huomattava on, että mallin tiedot tallentuvat vain, mikäli suurimman uskottavuuden estimaatit löydetään binäärisessä etsintäalgoritmissa, eikä ensimmäisessä vaiheessa, jossa estimoidaan rajat. Nämä arvot voidaan tarvittaessa kuitenkin generoida jälkikäteen draw-ohjelman avulla. ”Other”-välilehdellä on muita ohjelman toimintaan liittyviä kiinnostavia tietoja. Näitä ovat kiinnostavat pisteet (Points of interest), montako kierrosta etsintäalgoritmi teki (Design rounds), tieto siitä, loppuvatko jossain äänessä toistot kesken (Did sound run out) sekä etsintäalgoritmin kumman tahansa osan antamat kiinnostavat pisteet (POI by model).

Koeosan loputtua käyttäjä voi valita, jatketaanko kontrolliosaan vai lopetetaanko ohjelma. Kontrolliosassa toistetaan pelkästään ääniä, eikä vastetta enää monitoroida. Ääniä toistetaan sekunnin välein ja niiden järjestys on siis sama kuin koekäsittelyssä.

Pääohjelmalla on seitsemän aliohjelmaa: calculatePOI, drawFigure, optdesign, playSounds, playThese, randomizeSoundsLeft ja thetaest. Niiden tarkoituksena on laskea, sekoittaa ja soittaa ääniä, rekisteröidä vastauksia sekä piirtää kuvia. Aliohjelma thetaest laskee sijaintiparametrin θ ja leveysparametrin λ. Muita aliohjelmia käsitellään seuraavaksi tarkemmin.

4.2 Aliohjelmat

4.2.1 CalculatePOI

Alun perin binäärinen etsintäalgoritmi mittaa toistuvasti peräkkäin vasteen arvoja aina yhdessä prediktorin x arvossa kerrallaan. Ongelmana äänteiden havaitsemisen tutkimisen tapauksessa on, että samaa ääntä ei saa toistaa monta kertaa peräkkäin. Jos toistoja on monta peräkkäin, koehenkilön aivot eivät reagoi ärsykkeeseen samalla voimakkuudella, vaan reaktio vaimenee toistokertojen lisääntyessä. Tämän takia algoritmia piti muuttaa siten, että se mittaa vasteen arvoja useammassa prediktorin arvossa kerrallaan. Tämä mahdollistaa sen, että toistettavat äänet voidaan sekoittaa siten, että samaa ääntä ei toisteta kahta kertaa peräkkäin.

Aliohjelma calculatePOI (calculate points of interest) laskee kolme pistettä, joissa vasteen arvoa halutaan mitata ja sekoittaa näissä pisteissä tehtävät toistot. Aliohjelmalle viedään parametreina arvot x, a, b, *numberOfVoices* ja *n_inc*. Parametri x on joko jompikumpi mittausvälin päätepisteistä tai binäärisestä etsintäalgoritmista saatu prediktorin arvo. Parametrit *a* ja *b* määrittävät tutkittavan välin: a on prediktorin x suurin arvo, jossa vaste on saanut arvon 0 ja b on prediktorin *x* pienin arvo, jossa vaste on saanut arvon 1. Parametri *numberOfVoices* kertoo, montako ääntä kokeessa on kaiken kaikkiaan. Tässä tutkielmassa ääniä on 9 kappaletta. Parametri *n_inc* on sama kuin aiemmin eli yhden äänen toistokertojen määrä yhden estimointikerran aikana. Ohjelma palauttaa vektorin, jossa on toistettavat äänet satunnaisesti järjestettynä.

Aliohjelma laskee parametrin *x* läheisyydestä kaksi pistettä, joissa vasteen arvo halutaan mitata. Nämä kaksi pistettä sijaitsevat neljäsosan tarkasteltavasta välistä etäisyydellä pisteestä x eli ensimmäinen piste POI_a on kohdassa $x - ((b-a)/4)$ ja toinen piste POI_b on kohdassa $x + ((b-a)/4)$. Mikäli parametri x saa arvon 1, niin piste POI_a on kohdassa $x + 2(\mathrm{POI}_b - x)$ ja mikäli parametri x saa arvon *numberOfVoices*, niin piste POI_b on kohdassa $x - 2(x - \mathrm{POI}_a)$. Jos parametri x saa saman arvon kuin POI_a, niin $\mathrm{POI}_a = x - 1$. Jos parametri x saa saman arvon kuin POI_b, niin $\mathrm{POI}_b = x + 1$. Kaikki arvot pyöristetään lähimpään kokonaislukuun, koska parametri x voi saada vain kokonaislukuarvoja. Kun aliohjelma on laskenut arvot x, POI_a ja POI_b, nämä kolme pistettä tallennetaan vektoriin niin monta kertaa kuin parametri *n_inc* määrittelee. Tämän jälkeen vektorin sisältö sekoitetaan siten, että samaa arvoa ei ole kahdesti peräkkäin.

4.2.2 DrawFigure

Aliohjelman drawFigure tehtävänä on piirtää kuva mitatusta aineistosta ja sovittaa kuvaan logistisen kertymäfunktion kuvaaja. Aliohjelmalle viedään seuraavat parametrit: *xplot*, *xpoint*, *ypoint*, *beta_c*, *beta_c_found*, *beta_cIsFound* ja *numberOfSounds*. Parametri *xplot* sisältää tiedon x-akselin asteikosta. Parametreissa *xpoint* ja *ypoint* on vektorit mitatuille muuttujille x ja vasteille y. Parametrissa *beta_c* on estimoidut arvot parametreille a ja b, parametrissa *beta_c_found* on suurimman uskottavuuden estimaatit parametreille a ja b ja parametrissa *beta_cIsFound* on tieto siitä, joko suurimman uskottavuuden estimaatit on löydetty (0 = ei ole löytynyt). Parametri *numberOfSounds* sisältää tiedon parametrin x maksimiarvosta eli kokeessa käytettävien äänten lukumäärän.

Aliohjelma piirtää kuvan, jossa x-akselilla on ääniä vastaavien kokonaislukujen arvot ja y-akselilla on keskiarvo mitatuista vasteiden arvoista. Kuvaan piirretään sinisellä värillä kuvaaja binäärisen etsintäalgoritmin tuottamasta aineistosta ja vihreällä värillä kuvaaja kaikesta kokeen aikana kerätystä aineistosta. Sininen kuvaaja päivittyy pääohjelman ensimmäisen osan lopuksi ja jokaisen binäärisen etsintäalgoritmin tekemän kierroksen jälkeen. Vihreän kuvaaja piirretään ensimmäisen kerran, kun suurimman uskottavuuden estimaatit on löydetty (pääohjelman toisen osan lopussa) ja se päivittyy jokaisen äänen jälkeen.

Joissain tilanteissa kuvaaja ei välttämättä piirry ollenkaan tai pelkästään vihreä kuvaaja piirretään. Tällaisia tilanteita ovat esimerkiksi se, että estimaatit löydetään jo etsintäalgoritmin ensimmäisessä osassa ja se, että kerätystä aineistosta saadut arvot eivät täytä oletuksia. Tässä tilanteessa voidaan käyttää erillistä draw-ohjelmaa, joka pystyy excel-tiedoston pohjalta piirtämään kuvan myös kokeen jälkeen.

4.2.3 Optdesign

Aliohjelma optdesign on alun perin Juha Karvasen tekemä. Sen tehtävänä on laskea optimaalinen malli, kun vastemuuttuja on binäärinen. Tämän aliohjelma tekee binäärisen etsintäalgoritmin avulla. Aliohjelma on alun perin käyttänyt mallin estimointiin Gombertz-jakaumaa (kyseessä silloin logit mallin sijaan komplementaarinen log-log malli), mutta tätä tutkielmaa varten jakaumaksi on vaihdettu logistinen jakauma.

Aliohjelma tarvitsee kolme muuta aliohjelmaa toimiakseen: logistic_cdf, logistic_cov ja logistic_logl. Aliohjelma logistic_cdf laskee logistisen jakauman kertymäfunktion parametreilla a (skaalaparametri) ja b (sijaintiparametri). Aliohjelma logistic_cov laskee asymptoottisen kovarianssimatriisin logistisessa regressiossa parametreilla a ja b. Kovarianssi on havaittu Fisher-informaatiomatriisi käänteisenä. Aliohjelma logistic_logl puolestaan laskee logistisen uskottavuuden logistisessa regressiossa.

4.2.4 PlaySounds ja playThese

Ohjelmassa on kaksi eri ohjelmaa äänten toistamiseen, playSounds ja playThese, joista toinen on tarkoitettu koekäsittelyyn ja toinen on tarkoitettu kontrollikäsittelyyn. Aliohjelmaa playSounds käytetään koekäsittelyssä. Se soittaa koehenkilölle ääniä, lähettää triggerit EEG-laitteelle, kontrolloi vastausikkunaa ja lukee koehenkilön syötteen. Aliohjelmalle viedään seuraavat parametrit: *x*, *soundsLeft*, *a* ja *b*. Parametri *x* sisältää vektorin soitettavista äänistä, parametrissa *soundsLeft* on kaikkien äänten toistokertojen määrä ja parametrit *a* sekä *b* ovat parametreja vastausikkunassa olevalle symbolille.

Aluksi aliohjelma lukee äänet omiin muuttujiinsa. Äänten on oltava wav-tiedostoina (*Wafeform Audio File Format*) ja niiden bittinopeuden (*bit rate*) on oltava 1141 kbps. Seuraavaksi aliohjelma toistaa ääniä siinä järjestyksessä, kuin ne ovat vektorissa *x*. Ennen jokaisen äänen toistamista aliohjelma lähettää EEG-laitteelle triggerisignaalin. Triggerin tarkoituksena on tulosten analysointia varten merkata EEG-käyrään hetki, jolloin jokin tietty ääni on soitettu. Jokaisella äänellä on siis oma triggeri, joka koostuu rinnakkaisportin kautta EEG-laitteelle lähetettävistä tavuista. EEG-laitteessa portit muodostuvat binääriluvuista 1, 2, 4, 8, 16, 32, 64 ja 128. Näistä portit 64 ja 128 on varattu vasteiden lukemiseen, joten niitä ei tässä tutkielmassa tarvita. Kaikki triggerit väliltä 1-128 saadaan muodostettua binäärilukujen avulla.

Taulukko 1: EEG-laitteelle lähetettävät triggerisignaalit ja niitä vastaavat äänet.

Äänen numero	Triggeri
1	1000
2	0100
3	1100
4	0010
5	1010
6	0110
7	1110
8	0001
9	1001

Tässä tutkielmassa jokaisen äänen numeroa vastaa sama numero binäärimuodossa. Taulukossa 1 on ääniä vastaavat triggerisignaalit. Taulukosta voidaan huomata, että MATLAB ymmärtää binääriluvut käänteisessä järjestyksessä (vasemmalta oikealle) normaaliin esitystapaan nähden. Triggerisignaalin 0 (0000) tarkoituksena on nollata EEG-laite, jotta se pystyy rekisteröimään seuraavan triggerisignaalin. Nollaaminen tapahtuu ohjelman alussa ja aina äänen toistamisen

jälkeen. Äänen toistamisen jälkeen aliohjelma päivittää matriisia *soundsLeft* ja vähentää soitettua ääntä vastaavasta toistokertojen määrästä yhden.

Seuraavaksi ohjelma pitää sekunnin mittaisen tauon, jonka aikana koehenkilön kuulohavainnosta johtuva EEG-vaste ehtii tasoittua. Tämän jälkeen aliohjelma muuttaa vastausikkunan symbolin vihreäksi. Tämä on merkki koehenkilölle, että hän voi nyt vastata nappulaa painamalla. Kun koehenkilö on painanut nappulaa, symbolin väri muuttuu punaiseksi. Tämän jälkeen aliohjelma tallentaa, kumpaa nappulaa koehenkilö on painanut. Lopuksi aliohjelma palauttaa pääohjelmalle vektorit soitetuista äänistä ja niiden vasteista.

Aliohjelma playThese on tarkoitettu kontrollikäsittelyyn. Sille tarvitsee viedä parametrina ainoastaan äänten järjestyksen sisältävä vektori *x*. PlayThese lähettää EEG-laitteelle triggerit samalla tavalla kuin playSounds. Erona aliohjelmaan playSounds on, että playThese ei käsittele vasteita, vaan soittaa ääniä sekunnin välein. Se ei myöskään päivitä vastausikkunaa.

4.2.5 RandomizeSoundsLeft

Aliohjelmaa randomizeSoundsLeft tarvitaan, kun binäärinen etsintäalgoritmi on lopettanut suurimman uskottavuuden estimaattien etsimisen. Aliohjelmaa käytetään sen jälkeen, kun kiinnostavat pisteet on valittu ja toistojen määrä on päivitetty oikeaksi. Aliohjelman randomizeSoundsLeft tarkoituksena on sekoittaa kaikki jäljellä olevat äänet.

Aliohjelmalle viedään parametrina matriisi *soundsLeft*, jossa on tieto, montako kertaa kutakin ääntä tulee vielä toistaa. Aliohjelmassa on silmukka, joka laittaa ääniä satunnaisesti vektoriin, kunnes kaikki äänet on toistettu riittävän monta kertaa. Tämän jälkeen aliohjelma yrittää maksimissaan 200 kertaa sekoittaa äänet sellaiseen järjestykseen, että sama ääni soitetaan korkeintaan kahdesti peräkkäin. Sekoitusten määrän pitäisi olla tässä tapauksessa riittävä, koska testauksen aikana ohjelma joutui yrittämään sekoitusta yleensä noin 30 kertaa ennen onnistumistaan. Lopuksi aliohjelma palauttaa vektorin sekoitetuista äänistä pääohjelmalle.

5 Mittaukset ja ohjelman toiminta

5.1 Mittaukset

Mittauksia tehtiin yhteensä viitenä eri päivänä: 23.5, 30.5, 4.6, 5.6 ja 6.6.2014 ja ne suoritettiin psykologian laitoksen aivotutkimuslaboratoriossa Ylistönmäellä. Mittauksiin osallistui 18 henkilöä, joista 6 oli miehiä ja 12 naisia. Koehenkilöiden ikä vaihteli välillä 20–59 vuotta (keskiarvo 26.9 sekä moodi ja mediaani 25 vuotta). Kaikki paitsi yksi koehenkilöistä oli oikeakätisiä. 10 henkilöistä oli opiskelijoita ja 8 jo työelämässä. Humanististen alojen edustajia (kielten lukijoita) oli 7 kappaletta, kasvatustieteiden edustajia puolestaan oli 4. Loput olivat vaihtelevasti eri aloilta, joita olivat psykologia, filosofia, musiikki, biologia sekä sähkövoimatekniikka. Koehenkilöt kerättiin laittamalla yliopiston sähköpostilistojen kautta viesti, jossa haettiin koehenkilöitä. Vapaaehtoisiksi ilmoittautuneista valittiin ilmoittautumisjärjestyksessä koehenkilöt tutkimukseen. Tällöin kyseessä on niin sanottu opportunistinen otos (*opportunistic sample*).

Kokeen kesto oli noin 1-1.5 tuntia, riippuen ajasta, jonka valmistelut kestivät sekä mahdollisista ongelmista kokeen aikana tai ennen koetta. Itse koekäsittelyn kesto oli noin 30 minuuttia ja kontrollikäsittelyn kesto vajaan 10 minuuttia. Valmisteluihin kuului taustatietojen kysely ja EEG-myssyn laittaminen paikoilleen. EEG-mittauksissa käytettiin Brain Products:in QuickAmp-vahvistinta ja näytteenottotaajuus oli 1000Hz. Mittaukset tehtiin 21-elektrodisella myssyllä, lisäksi otsaan kiinnitettiin erillinen maadoituselektrodi. Elektrodeihin laitettiin sähkönjohtavuutta parantavaa geeliä, jonka jälkeen myssy aseteltiin elektrodit päätä vasten. Myssy kiinnitettiin ristikkäin vedettävillä nauhoilla koehenkilön rintakehän ympärille laitettuun nauhaan.

Kuva 3: Koetilanne. Koehenkilöllä on EEG-myssy päässä, kuulokkeet korvilla, hiiri sylissä ja ruudulla näkyy vastausikkuna.

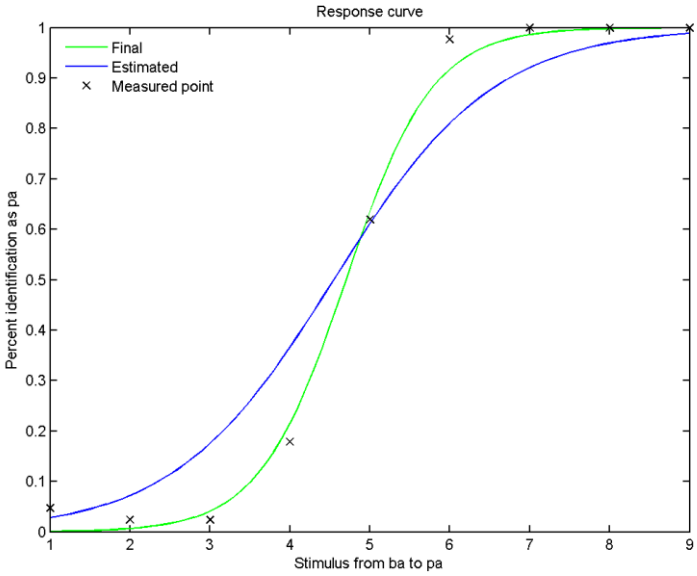
Koehenkilöt saivat ohjeet koetta varten, heille annettiin hiiri syliin vastaamista varten ja heille asetettiin kuulokkeet korville. Koehenkilöt istuivat tuolissa, jonka edessä oli tietokoneen näyttö. Tietokoneen näytöllä näkyi heille pelkästään ohjelman vastausikkuna, jota heidän täytyi äänen kuullessaan klikata joko hiiren oikealla tai vasemmalla painikkeella antaakseen vastauksensa. Kuvassa 3 näkyy koehenkilö valmiina mittaukseen.

Kaikki mittaukset eivät sujuneet aivan ongelmitta, esimerkiksi EEG-laite aiheutti muutaman kerran ongelmia. Laitteen liitin rikkoutui kerran, jolloin EEG-vasteisiin aiheutui häiriötä (korjautui, kun vaihdettiin liitin). Lisäksi triggereiden lähetys ei jostain syystä parina kertana onnistunut (käynnistettiin laitteet uudelleen ja tarkistettiin liitännät, jolloin ongelma korjautui). Myös elektrodit aiheuttivat häiriöitä joillain koehenkilöillä kesken kokeen. Tämä pyrittiin korjaamaan asentamalla maaelektrodi uudestaan sekä parantamalla elektrodien asentoa.

Koehenkilöiden yleinen ilmapiiri oli positiivinen ja he olivat kärsivällisiä, vaikka ongelmia välillä ilmaantui. Osa koki vaikeaksi äänteiden erottelemisen toisistaan, osa ihmetteli äänteissä ollutta /pha/ äännettä (joka johtui englanninkielisille tarkoitetuista äänistä) ja jotkut huomasivat painaneensa välillä väärää nappulaa. Muutama sanoi tunnistaneensa, että äänteitä on useampia erilaisia ja pari sanoi, että äänet kuulostavat englanninkielisiltä. Yksi henkilö sanoi myös, että hänellä oli vaikeuksia päättää, kuuluuko hänen vastata suomenkielen vai englanninkielen äänteiden mukaisesti.

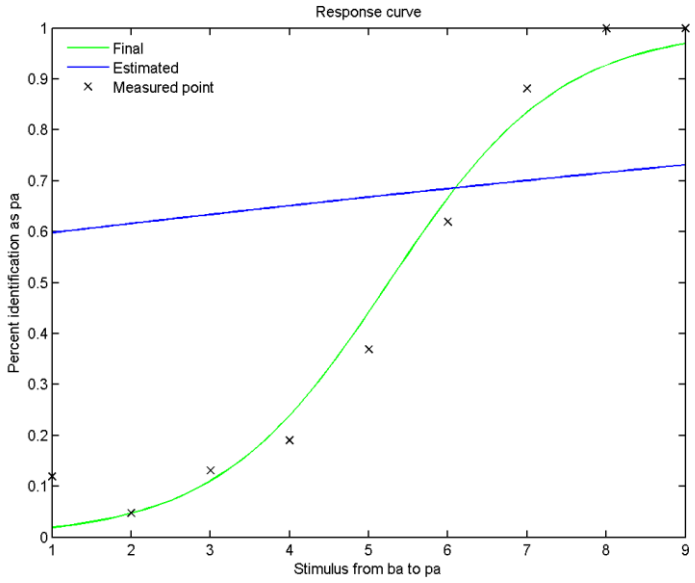
5.2 Ohjelman toiminnan analysointi

Tulokset vaihtelivat paljon eri koehenkilöiden välillä. Kaikkien koehenkilöiden kuvaajat löytyvät liitteistä ja niiden yhteyteen on kirjattu tiedot valituista pisteistä ja kuvista katsotuista oikeista pisteistä. Tähän on valittu vain esimerkkikuvaajia erityyppisistä tapauksista. Parin koehenkilön kohdalla ohjelma käyttäytyi odotusten mukaisesti. Esimerkiksi kuvassa 4 nähdään, että algoritmin avulla estimoitu kuvaaja näyttää hyvin samanlaiselta verrattuna kaikista toistoista piirrettyyn kuvaajaan. Tässä tapauksessa etsintäalgoritmin tuottamasta mallista saatiin estimaateiksi äänet 3 ja 5, jolloin kiinnostaviksi pisteiksi valittiin 3,4 ja 5. Kuvan perusteella pisteet 4 ja 5 pitävät paikkansa ja joko 3 tai 6 olisi voinut kolmas kiinnostava piste. Samanlaisia tuloksia on kahdessa muussa vastaavassa tapauksessa.



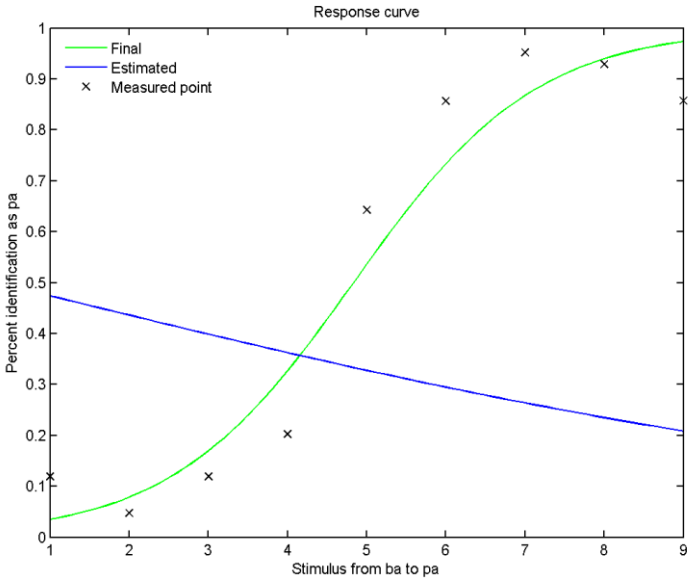
Kuva 4: Odotusten mukaiset tulokset.

Monella koehenkilöllä puolestaan etsintäalgoritmin tuottama kuvaaja on melkein vaakasuora, joskin kasvava. Kuitenkin kaikista toistoista piirretty kuvaaja on oletusten mukainen eli s-käyrä. Esimerkiksi kuvassa 5 näkyy tilanne, jossa lopputulokset ovat mallin mukaiset, mutta etsintäalgoritmin tulos ei ole. Tässä tapauksessa etsintäalgoritmi on estimoinut pisteet 1,3,5 ja 7, jolloin kiinnostaviksi pisteiksi on valittu pisteet 3,4 ja 5. Kuvasta kuitenkin näkee, että oikeampi valinta olisi ollut 6 eikä 3. Toisaalta tällaisessa tapauksessa, jossa etsintäalgoritmi antaa 4 arvoa, on vaikea päätellä oikeat kolme pistettä. Muissa vastaavissa tapauksissa, kun etsintäalgoritmi on estimoinut kaksi pistettä, valitut kiinnostavat pisteet ovat olleet oikeat pisteet. Puolestaan vastaavissa tapauksissa, joissa pisteitä on estimoitu neljä etsintäalgoritmissa, kaksi on ollut oikein ja kolmas väärä tai epävarma.



Kuva 5: Tulos, joissa estimointi tuottaa hieman kasvavan, vaakasuoran kuvaajan ja lopulliset tulokset odotusten mukaisen käyrän.

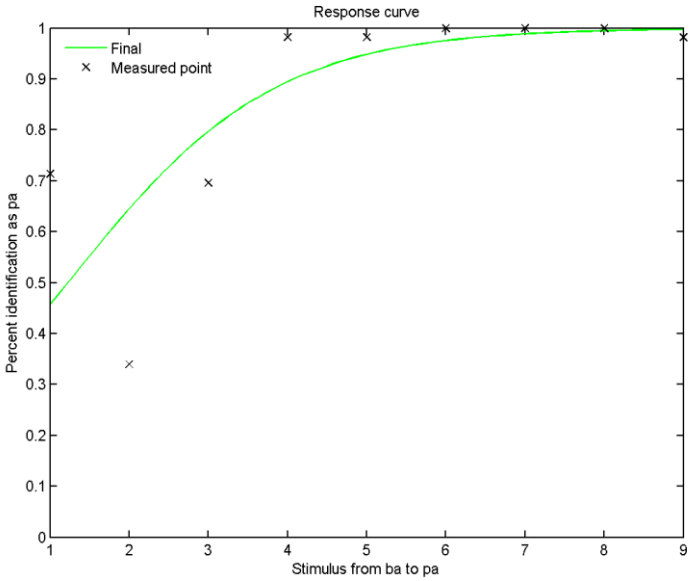
Joillain koehenkilöillä puolestaan etsintäalgoritmi estimoi suunnilleen lineaarisen, laskevan suoran, vaikka kaikista toistoista piirretty kuvaaja näyttää aikalailla odotusten mukaiselta. Kuvassa 6 on kyseessä tällainen tapaus. Tässäkin tapauksessa etsintäalgoritmista on saatu tuloksena pisteet 1, 3, 5, ja 7, jolloin kiinnostaviksi pisteiksi on valittu 3, 4 ja 5. Tässä tapauksessa on hankala sanoa, kumpi luvuista, 3 vai 6, on oikeasti se kolmas kiinnostava piste. Kahdessa vastaavassa tapauksessa etsintäalgoritmi on löytänyt viisi pistettä. Tällöin kiinnostavia pisteitä ei valita vaan kaikkia ääniä toistetaan saman verran. Näissä tapauksissa kuvaajien perusteella on hankala sanoa, mitkä ovat oikeat kiinnostavat pisteet, koska vaihtoehtoja on useita.



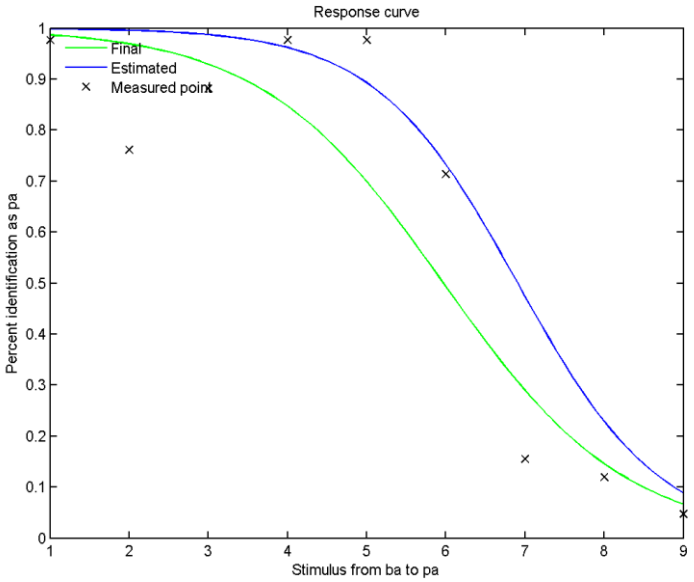
Kuva 6: Tulos, joissa estimointi tuottaa laskevan lineaarisen kuvaajan ja lopullinen kuvaaja muistuttaa suunnilleen odotuksia.

Seuraavaksi on tuloksia, joissa etsintäalgoritmi ei ole pystynyt estimoimaan yhtäkään pistettä. Tämä johtuu siitä, että kaikki etsintäalgoritmin saamat vastaukset ovat olleet samoja (kaikki pelkkiä ykkösiä). Samankaltainen tapaus on sellainen, jossa etsintäalgoritmi on saanut estimoiduksi vain yhden päätypisteen. Tällaisia tapauksia on yhteensä kolme kappaletta ja ne ovat tuloksiltaan kovin erilaiset. Kuvassa 7 näkyy tulos, joka ei ole ollenkaan odotusten mukainen, koska äänestä 4 eteenpäin vastaukset ovat olleet ykköstä ja äänten 1,2 ja 3 kohdalla vastauksia on tullut sekaisin. Oikeiden pisteiden valinta olisi ollut etsintäalgoritmin avulla mahdoton tehtävä, koska alussa kaikki vastaukset olivat pelkkää ykköstä. Toisessa tapauksessa kiinnostavat pisteet ovat kaikista toistoista piirretyn kuvaajan perusteella 4 ja 5 sekä joko 3 tai 6. Kolmannessa tapauksessa puolestaan etsintäalgoritmi on löytänyt pisteen 9, jolloin kiinnostaviksi pisteiksi on valittu 7,8 ja 9. Oikeista arvoista on tässä tapauksessa hankala sanoa, koska kuvaaja on kasvava lineaarinen suora s-käyrän sijaan.

Kahden koehenkilön vastaukset puolestaan olivat täysin päinvastaiset oletukseen verrattuna eli äänne /ba/ tunnistettiin äänteeksi /pa/ ja äänne /pa/ tunnistettiin äänteeksi /ba/. Kuvassa 8 näkyy toinen näistä tapauksista. Luultavimmin tällaiset tulokset johtuvat siitä, että koehenkilö on painanut vahingossa nappuloita väärinpäin. Ohjelma ja kiinnostavien pisteiden valinta kuitenkin toimii myös tällaisessa tapauksessa. Kuvan 8 kohdalla etsintäalgoritmi on löytänyt estimaatiksi pisteen 9, jolloin kiinnostaviksi pisteiksi on valikoitunut 7,8 ja 9. Kuvan perusteella oikeammat vaihtoehdot olisivat kenties olleet 6,7 ja 8.

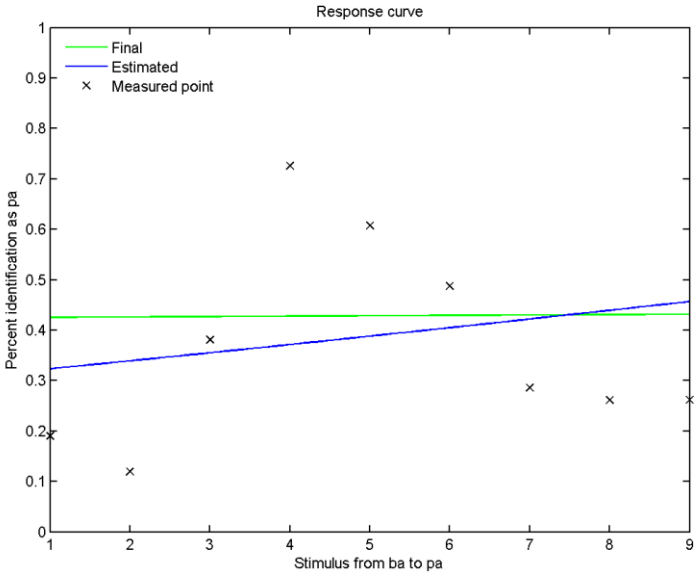


Kuva 7: Tapaus, jossa etsintäalgoritmi ei ole pystynyt estimoimaan yhtäkään pistettä ja lopulliset tuloksetkin ovat erilaiset kuin odotettiin.



Kuva 8: Tulos, jossa äänteet on tulkittu väärinpäin.

Yhdellä koehenkilöllä puolestaan tulokset ovat erikoiset, koska sekä estimoitu että kaikista toistoista laskettu kuvaaja ovat melkein vaakasuoria. Pisteitä katsomalla myös näkee, että tässä tapauksessa logistisen regression tuottama s-käyrä ei ole sopiva mallintamaan vasteita. Vasteen arvot nimittäin ensin kasvavat kohti puoliväliä, jonka jälkeen ne laskevat, jolloin kyseeseen tulisi tässä tapauksessa ennemminkin paraabelin muotoinen käyrä. Tämä saattaa johtua siitä, että englanninkielisten äänten takia äänne /pha/ on tulkittu tarkoittamaan äännettä /ba/ eikä äännettä /pa/, kuten tarkoitus oli.



Kuva 9: Tulos, joka eroaa kaikilta osin odotuksista.

Liitteeseen 1 on koottu kaikkien 18 koehenkilön kuvat. Ohjelma on valinnut kiinnostavat pisteet oikein 7 tapauksessa. Oikeaksi on tulkittu myös sellaiset tilanteet, joissa kaksi pistettä ovat oikein ja kolmannesta pisteestä ei pystytä sanomaan varmuudella, mikä olisi oikea valinta. 4 tapauksessa kiinnostavista pisteistä on valittu oikein kaksi kolmesta. Tapauksia, joista ei voi tulkita oikeita pisteitä on 5 kappaletta ja kokonaan väärin menneitä on kaksi kappaletta. Väärin menneistä kumpikin tapaus on sellainen, että ohjelma ei ole pystynyt määrittämään yhtäkään kiinnostavaa pistettä. Kokonaisuudessaan siis 11 tapauksessa ohjelma suoriutui hyvin, 5 tapauksessa ei pystytä sanomaan toimiko ohjelma vai ei ja 2 tapauksessa ohjelma toimi huonosti.

Useimmissa tapauksissa ohjelma ei ehtinyt missään vaiheessa etsintäalgoritmin toiseen vaiheeseen vaan löysi kaksi tai useampia pisteitä jo ensimmäisessä vaiheessa, jossa tutkitaan ylä- ja alarajoja sekä niiden läheisyydestä pisteitä (äänet 1, 3, 5, 7 ja 9). Ehdottomasti ohjelman kompastuskivi on se, että koehenkilöt saattavat virheellisesti painaa jonkin vastauksen väärin, jolloin löydetään väärät estimaatit. Virheiden todennäköisyys oli suurempi nimenomaan ohjelman alussa, koska tehtävä oli koehenkilöille uusi, eikä heillä ollut minkäänlaista harjoitteluosiota ennen varsinaisen kokeen aloittamista. Oletettavaa on, että kokeen aikana koehenkilöille kehittyi parempi rutiini vastaamiseen, jolloin harjoitteluosiosta olisi ollut hyötyä ja se olisi saattanut parantaa ohjelman tuloksia.

Ohjelmakoodi itsessään toimi melko moitteettomasti ja ohjelman toiminnassa havaittiin mittausten aikana vain pari bugia. Ensimmäinen niistä oli, että muutamalla henkilöllä ohjelma ei pystynytkään enää toistamaan ääniä ja kaatui. Tämä siis tapahtui kesken kokeen. Kahdella koehenkilöllä koe aloitettiin kokonaan alusta tämän virheen takia. Yhden koehenkilön kanssa ehdittiin tehdä 487 toistoa, joten päädyin tallentamaan tulokset manuaalisesti, jonka jälkeen tehtiin kontrollikäsittely näille 487 toistolle. Virheilmoituksena oli, että äänilaitteeseen ei

saada yhteyttä. Epäiltiin, että tämä saattaa johtua MATLAB:in äänentoistofunktiosta, mikäli se yrittää streamata suoraan sen sijaan, että se lataisi äänet valmiiksi puskuriin. Vika saattoi myös johtua ohjelmiston ja laitteiden huonosta yhteensopivuudesta, ajureista tai jostain teknisestä viasta. Tämä vika ei selvinnyt, joten ohjelmaan lisättiin try-catch-silmukka, joka ottaisi kiinni mahdolliset virheet ja tallentaisi siihen mennessä kerätyt tulokset. Toinen bugi johtui käyttäjien syötteen ja mallin oletusten yhteensopimattomuudesta. Tämä tapahtui vain yhden henkilön kohdalla ja ongelma korjattiin myös tässä tapauksessa try-catch-silmukalla.

6 Pohdintaa

Tässä pro gradu -tutkielmassa tarkasteltiin mukautuvan mallin soveltamista kategorisen havaitsemisen tutkimiseen kognitiivisessa neuropsykologiassa. Kategorista havaitsemista tutkittiin soittamalla 18 koehenkilölle äänteitä /ba/ ja /pa/ sekä näiden välimuotoja. Koehenkilöiden tehtävänä oli koekäsittelyssä vastata, kumpi äänne heidän mielestään oli kyseessä. Kontrollikäsittelyssä tehtävänä puolestaan oli pelkästään kuunnella äänteet samassa järjestyksessä kuin koekäsittelyssä.

Jokaiselle koehenkilölle muodostettiin MATLAB:lla tehdyllä ohjelmalla kaksi havaitsemiskategoriaa binäärisen etsintäalgoritmin avulla. Ensimmäinen kategoria oli ”kiinnostavat äänet”, joihin kuuluivat ne kolme ääntä, jotka koehenkilön oli vaikein luokitella kategoriaan /ba/ tai /pa/. Toinen kategoria oli ”ei-kiinnostavat äänet”, johon kuuluivat loput 6 ääntä. Kiinnostavia ääniä soitettiin yhteensä saman verran kuin ei-kiinnostavia ääniä, jolloin jokaista kiinnostavaa ääntä soitettiin kaksinkertainen määrä verrattuna ei-kiinnostavaan ääneen. Mikäli kiinnostavia ääniä ei pystytty määrittämään, toistettiin kaikkia ääniä satunnaisesti saman verran.

11 koehenkilön tapauksessa ohjelma toimi hyvin, eli se valitsi kiinnostavien äänteiden kategorian oikein tai yhden äänen verran pieleen. Valinta tulkittiin oikeaksi myös sellaisessa tapauksessa, että yhtä kolmesta oikeasta äänestä ei ole jälkikäteen pystytty määrittämään kuvasta yksiselitteisesti. 5 tapauksessa ei ole jälkikäteen pystytty tulkitsemaan, mitkä ovat oikeat pisteet ja 2 tapauksessa valitut pisteet ovat kokonaan väärin. Yleisesti ottaen ohjelma toimi siis kohtuullisen hyvin.

Ohjelman heikkoutena oli ehdottomasti koehenkilöiden vastauksista johtuva epävarmuus. Esimerkiksi koehenkilöiden vahinkopainallukset saattoivat aiheuttaa sen, että algoritmi löysi vääriä pisteitä kiinnostaviksi pisteiksi. Tulevaisuudessa ohjelmaa olisi tarkoitus muokata siten, että vaste ei riippuisikaan koehenkilöiden vastauksista vaan esimerkiksi aivosähkökäyrästä saatavista tuloksista. Tällainen vasteen sitominen fyysiseen mittariin poistaisi koehenkilöistä johtuvat virheellisyydet ja epävarmuudet.

Tulevaisuudessa kannattaisi myös miettiä, voisiko havaitsemiskategorioiden suuruutta muuttaa. Tämän tutkielman tuloksista voidaan nimittäin huomata, että kaikille koehenkilöille kolmen kiinnostavan äänen kategoria ei välttämättä ole optimaalisin, vaan joillakin saattoi olla kaksi tai neljä kiinnostavaa ääntä. Yksi mahdollisuus olisi käyttää aineistosta estimoitua sijaintiparametria λ ja käyrän leveyttä kuvaavaa parametria θ apuna kategorioiden määrittämisessä. Tällaisessa mallissa toki ongelmaksi voi nousta mahdollisen valinta-algoritmin monimutkaisuus. Monimutkainen algoritmi on hankalampi toteuttaa ja lisäksi se kasvattaa virheherkkyyttä ja vaikeuttaa virheiden löytämistä.

Seuraava looginen askel olisi muokata ohjelmaa toimimaan siten, että toistojen määrää optimoitaisiin paitsi kategoriakohtaisesti (kiinnostavia ääniä toistetaan enemmän kuin muita) niin myös kokonaismäärältään. Tällä hetkellähän

kaikille koehenkilöille suoritetaan kiinteä määrä toistoja (504 kappaletta). Kuitenkin optimaalisinta olisi, että jokaiselle koehenkilölle soitettaisiin toistoja vain sen verran kuin on tarpeellista. Mittarina toistojen tarpeelliselle määrälle voitaisiin käyttää esimerkiksi sitä, että toistoja tehtäisiin niin paljon kuin analyysit edellyttävät. Tällöin koe voitaisiin lopettaa, kun aineistoa on kerätty riittävästi analyysien suorittamista varten.

Mukautuva malli näyttäisi olevan hyödyllinen menetelmä kategorisen havaitsemisen tutkimiseen. Tämän asian puolesta puhuu se fakta, että aiemmissa tutkimuksissa yksilöiden väliset erot on yleensä sivuutettu niputtamalla kaikki koehenkilöt samoihin, ennalta määrättyihin havaitsemiskategorioihin. Tässä tutkielmassa pyrittiin kehittämään menetelmä, joka huomioisi koehenkilöiden yksilölliset erot havaitsemisessa ja ohjelmasta saatiinkin melko virheetön ja toimiva. Lisäksi ohjelma toimii hyvänä pohjana myöhemmälle kehittämiselle.

Lähteet

Abdelbasit K. & Plackett R. (1983). Experimental designs for binary data. *Journal of American Statistical Association*, vol.7 8, No. 381, 90–98.

Albert A. & Anderson J.A. (1984). On the existence of maximum likelihood estimates in logistic regression models. *Biometrika*, vol. 71, 1-10.

Altman, Y. (2012). cprintf, MATLAB-function.
URL: http://www.mathworks.com/matlabcentral/fileexchange/24093-cprintf-display-formatted-colored-text-in-the-command-window
Viitattu 1.8.2014.

Atkinson & Donev & Tobias (2007). *Optimum experimental designs with SAS*. Oxford University Press, Oxford.

Beaumont, J. G. (2008). *Introduction to neuropsychology* (2nd edition). The Guilford Press, New York.

Czepiel S. (2002). *Maximum Likelihood Estimation of Logistic Regression Models: Theory and Implementation*.
[verkkojulkaisu]: URL http://czep.net/stat/mlelr.html. Viitattu 15.5.2013.

Harley T. (2001). *The Psychology of language: From data to theory* (2nd edition), Psychology Press, Hove, East Sussex.

Jeans J. (1968). *Science & music*. The Macmillan Company, New York.

Karvanen J., Vartiainen J., Timofeev A. & Pekola J. (2007). Experimental designs for binary data in switching measurements on superconducting Josephson junctions. *Journal of the Royal Statistical Society: Series C (Applied Statistics),* vol. 56, No. 2, 167–181.

Karvanen J. (2008). Efficient initial designs for binary response data. *Statistical Methodology*, vol. 5, 462–473.

Karvanen J. (2008). Optdesign, MATLAb-package.
URL: http://www.blackwellpublishing.com/rss/Readmefiles/56p2Karvanen.htm
Viitattu 20.3.2013.

Konis K. (2007). *Linear programming algorithms for detecting separated data in binary logistic regression models*. DPhil Thesis. Worcester College, University of Oxford.

Liberman A., Harris K., Hoffman H. & Griffith B. (1957). The discrimination of speech sounds within and across phoneme boundaries. *Journal of Experimental Psychology*, vol. 54, 358–368.

MATLAB (R2011a), The MathWorks Inc, Massachussetts.

Minkin S. (1987). Optimal designs for binary data. *Journal of the American Statistical Association,* vol. 82, 1098–1103.

Monahan J. (2001). *Numerical Methods of Statistics*. Cambridge University press, New York.

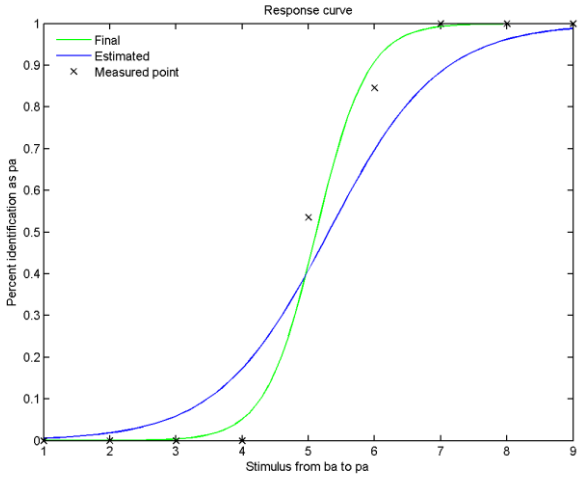
Silvapulle M. (1981). On the existence of maximum likelihood estimators for the binomial response models. *Journal of Royal Statistical Society,* vol.43. no.3, 310-313.

Syed M. & Kotsireas I. & Pardalos M. (2011). D-Optimal designs: A Mathematical programming approach using cyclotomic cosets. *Informatica*, vol. 22, no.4, 577–587.

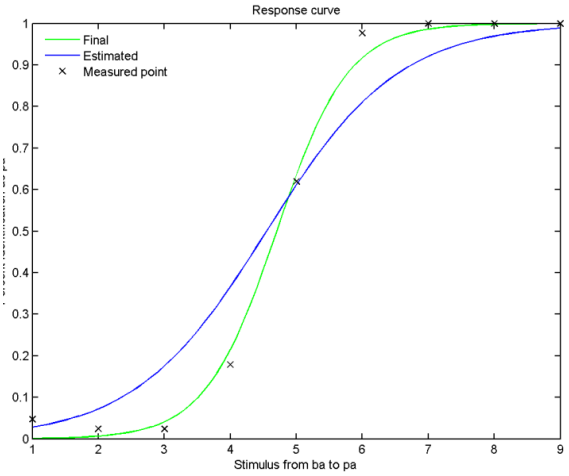
Liitteet

Liite 1: Ohjelman tuottamat kuvaajat

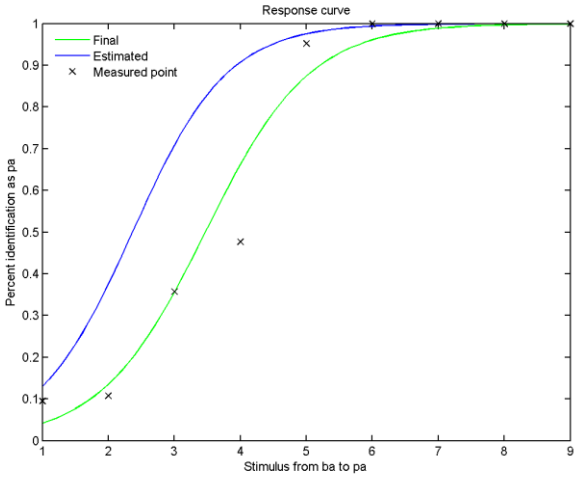
Jokaisen kuvan alle on listattu etsintäalgoritmilla estimoidut pisteet, ne pisteet, jotka ohjelma on valinnut kiinnostaviksi pisteiksi sekä silmämääräisesti määritellyt 3 niin sanottua oikeaa pistettä. Jos kuvasta ei pysty ollenkaan sanomaan, mitkä ovat kolme kiinnostavaa pistettä, niin tähän kohtaan on kirjattu ”Ei voida sanoa”. Kuvat ovat karkeasti samassa järjestyksessä kuin niitä on luvussa 5 käsitelty.



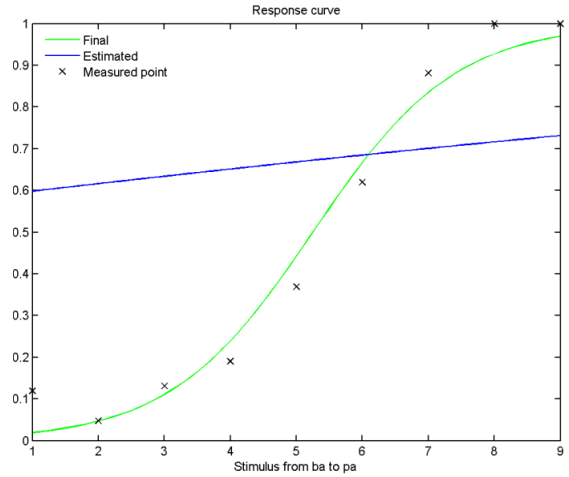
Etsintäalgoritmin antamat pisteet: 5
Valitut pisteet: 4,5 ja 6
Oikeat pisteet: 5,6 ja joko 4 tai 7



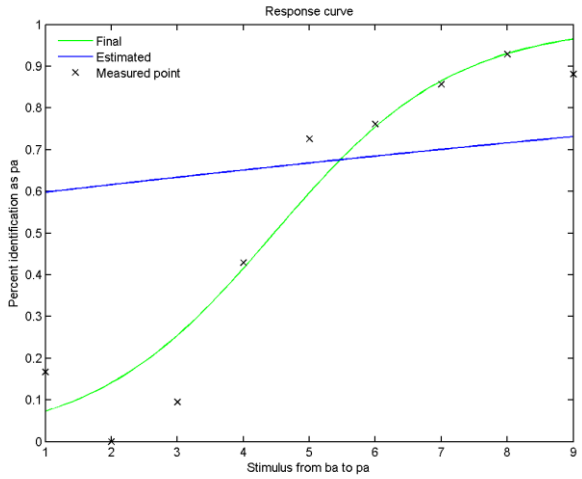
Etsintäalgoritmin antamat pisteet: 3,5
Valitut pisteet: 3,4 ja 5
Oikeat pisteet: 4,5 ja joko 3 tai 6



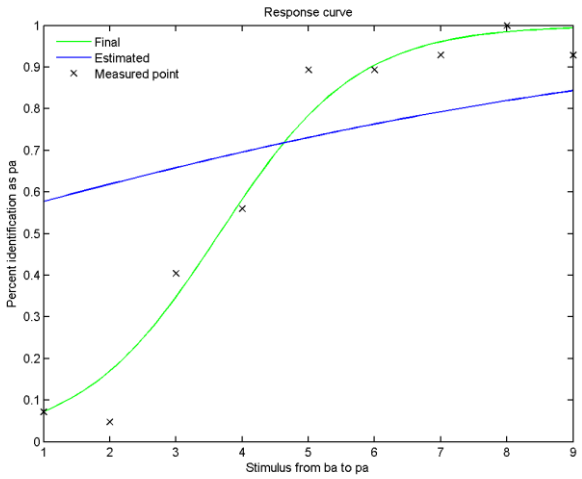
Etsintäalgoritmin antamat pisteet: 1,2 ja 3
Valitut pisteet: 1,2 ja 3
Oikeat pisteet: 2,3 ja 4



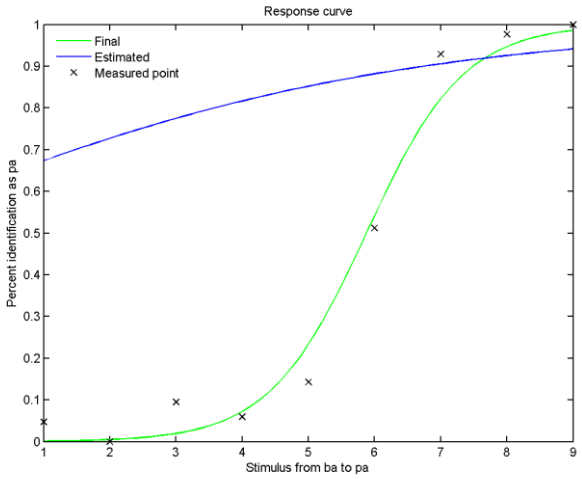
Etsintäalgoritmin antamat pisteet: 1,3,5 ja 7
Valitut pisteet: 3,4 ja 5
Oikeat pisteet: mahdollisesti 4,5 ja 6



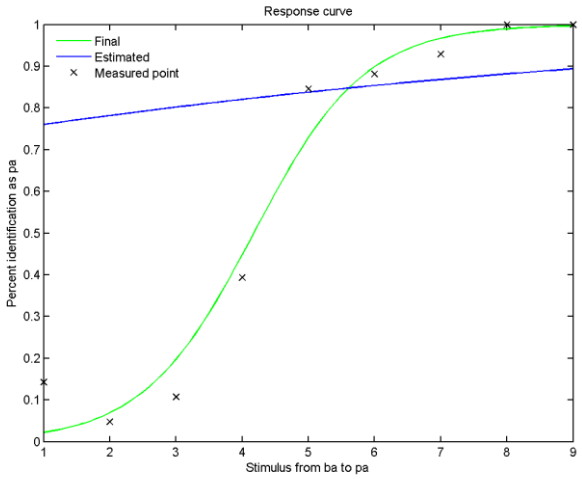
Etsintäalgoritmin antamat pisteet: 1,3,7 ja 9
Valitut pisteet: 4,5 ja 6
Oikeat pisteet: 4,5 ja 3 tai 6



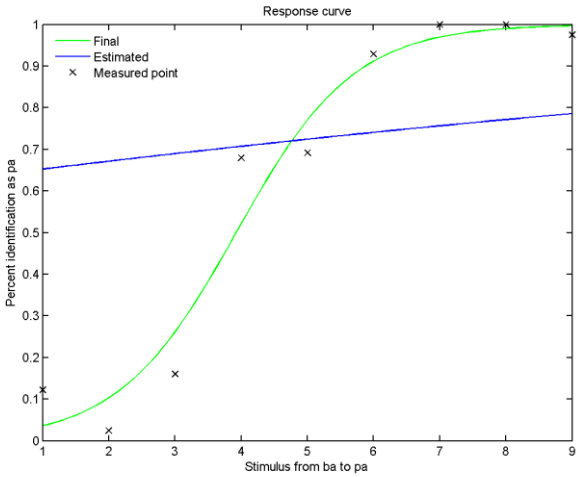
Etsintäalgoritmin antamat pisteet: 1,3,7 ja 9
Valitut pisteet: 4,5 ja 6
Oikeat pisteet: 3,4 ja 2 tai 5



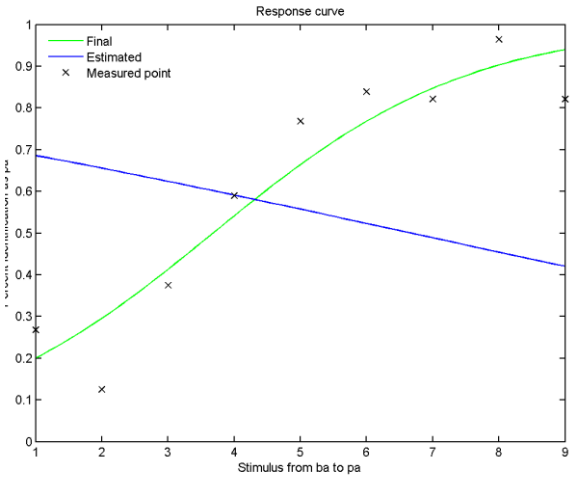
Etsintäalgoritmin antamat pisteet: 1 ja 5
Valitut pisteet: 4,5 ja 6
Oikeat pisteet: 5,6 ja 4 tai 7



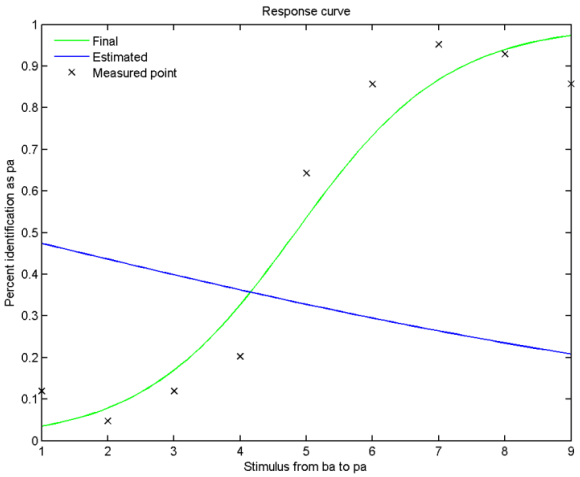
Etsintäalgoritmin antamat pisteet: 3 ja 5
Valitut pisteet: 3,4 ja 5
Oikeat pisteet: 4,5 ja 3 tai 6



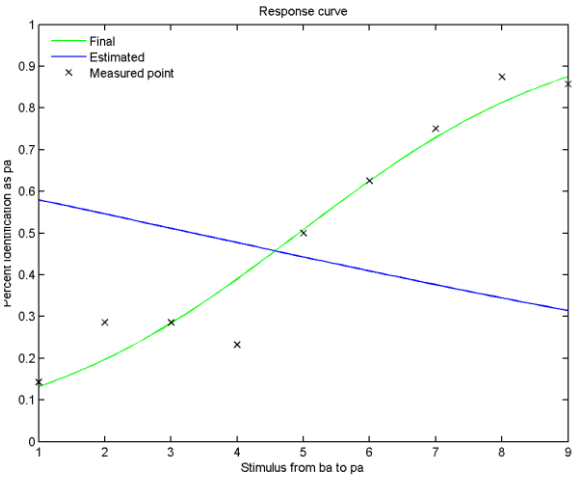
Etsintäalgoritmin antamat pisteet: 3 ja 5
Valitut pisteet: 3,4 ja 5
Oikeat pisteet: 4,5 ja 3 tai 6



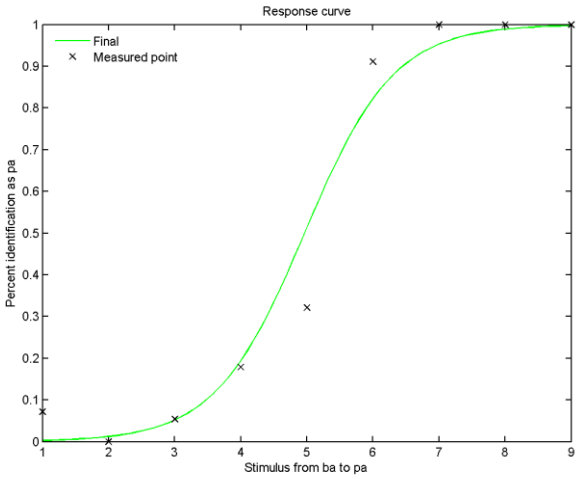
Etsintäalgoritmin antamat pisteet: 1,3,5,7 ja 9
Valitut pisteet: Ei yhtään
Oikeat pisteet: Ei voida sanoa



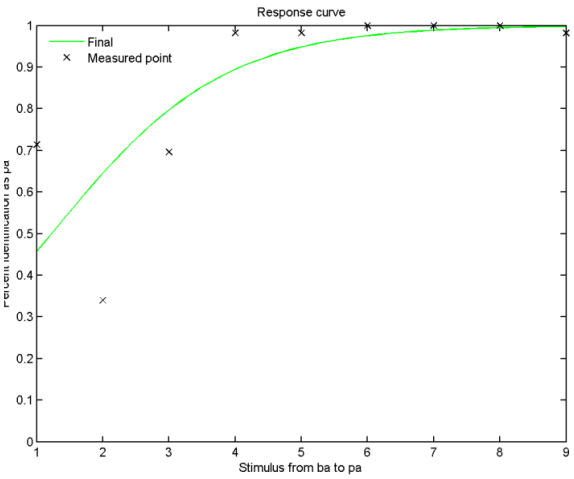
Etsintäalgoritmin antamat pisteet 1,3,5 ja 7
Valitut pisteet: 3,4 ja 5
Oikeat pisteet: 4,5 ja 3 tai 6



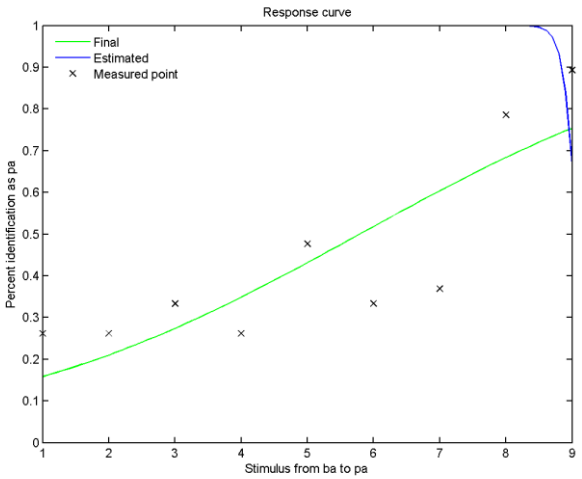
Etsintäalgoritmin antamat pisteet: 1,3,5,7 ja 9
Valitut pisteet: Ei yhtään
Oikeat pisteet: Ei voida sanoa



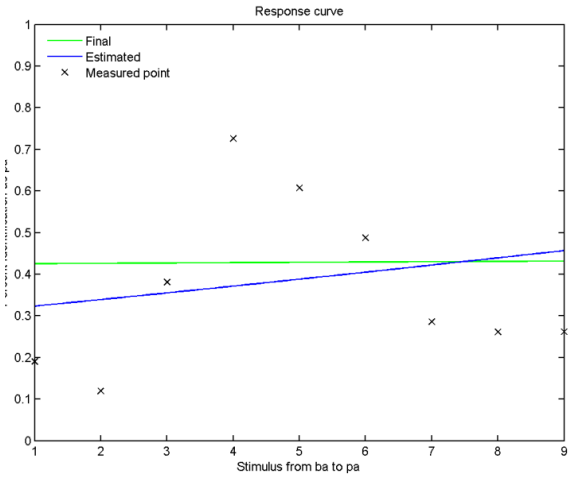
Etsintäalgoritmin antamat pisteet: Ei yhtään
Valitut pisteet: Ei yhtään
Oikeat pisteet: 4,5 ja 3 tai 6



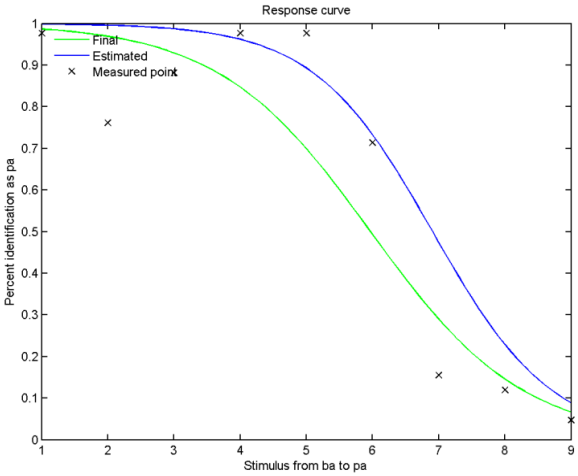
Etsintäalgoritmin antamat pisteet: Ei yhtään
Valitut pisteet: Ei yhtään
Oikeat pisteet: 1,2 ja 3



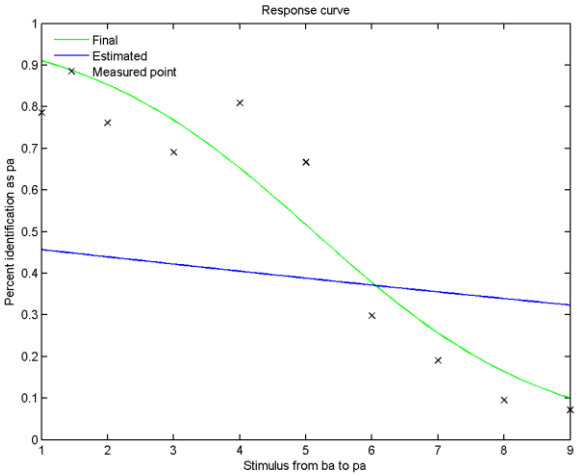
Etsintäalgoritmin antamat pisteet: 9
Valitut pisteet: 7,8 ja 9
Oikeat pisteet: Ei voida sanoa



Etsintäalgoritmin antamat pisteet: 3,5 ja 7
Valitut pisteet: 4,5 ja 6
Oikeat pisteet: Ei voida sanoa



Etsintäalgoritmin antamat pisteet: 9
Valitut pisteet: 7,8 ja 9
Oikeat pisteet: 6,7 ja 8



Etsintäalgoritmin antamat pisteet: 1,5 ja 7
Valitut pisteet: 4,5 ja 6
Oikeat pisteet: Ei voida sanoa

Liite 2: Esimerkit excel-tallennuksista

Experiment-välilehti (20 ensimmäistä arvoa):

Sound	Response
5	0
1	0
3	1
1	0
3	0
5	1
3	0
5	1
1	0
5	1
9	1
5	0
7	1
9	1
7	1
5	0
7	1
9	1
9	1

Control-välilehti (20 ensimmäistä arvoa):

Sound
5
1
3
1
3
5
3
5
1
5
9
5
7

9
7
5
7
9
9

Other-välilehti:

Points of interest	Design rounds	Did sound run out	POI by model
3	1	No	3
4			5
5			

Statistics-välilehti:

Parameter	Estimate	Standard deviation	Final Estimate	Final Standard deviation
a	1,000179	0,559859	1,849867	0,146507
b	-4,55082	2,709122	-8,69554	0,715684
theta	5,17235	0,44637	5,037119	0,053172
lambda	0,185937	0,10408	0,100532	0,007962

Liite 3: Koehenkilöiltä kerätyt taustatiedot

1. Nimi
2. Sukupuoli
3. Ikä
4. Ala, jolla opiskelee/työskentelee
5. Kätisyys: onko oikea vai vasenkätinen?

Liite 4: Pääohjelman MATLAB-koodi

```
%Program to test categorical perception. This program is designed to play
%sounds to test subject and register their responses. Responses are binary
%which means that there are only two alternative answering options.
%
%Program calculates the optimal desing for binary response data using
%logistic distribution (function optdesign.m). When maximum likelihood
%estimates haven't been found, program uses optdesign to calculate sounds
%to be played. When maximum likelihood estimates are found program
%selects three points that are more interesting than others and plays them
%twice as much as other sounds.
%
%Program generates two windows, one for response curve and one for
%answering. Window for answering is for test subject and it has green
%symbol when it is possible to answer and it has red symbol otherwise. The
%response curve window tries ti draw the response curve after every estimation.
%If it fails, the window will be blank.
%
%Program saves gathered data into an excel-sheet. Data will consist order
%of sounds, responses, estimates, three most interesting points and other
%info about functionality of the program. It will also save the
%response curve as a png-file.
%
%Uses:        calculatePOI.m
%             drawFigure.m
%             optdesign.m
%             playSounds.m
%             playThese.m
%             randomizeSoundsLeft.m
%                  thetaest.m
%
%Author: Minna Lehtomäki
%Version 1.0 2013-26-03
%Version 2.0 2014-15-05
%Version 3.0 2014-18-05
%Version 4.0 2014-20-05
%

clear all
minlimit=1;           %minimum limit for x
maxlimit=9;           %maximum limit for x
t = (1/16:1/8:1)'*2*pi;
a = sin(t);
b = cos(t);
answer = false;
number = int2str(round(1 + (100-1) * rand));
while (answer == false)
   result = input('Do you want to run the experiment?\nWrite y/n (yes/no) to continue:\n>>', 's');
   switch result
      case 'y'
         answer = true;
      case 'n'
         answer = true;
         on=false;
         error('');
      otherwise
```

```
            cprintf('err','INCORRECT INPUT');
            cprintf('text', ', please try again.\n\n');
        end
    end
    if strcmpi('n',result) == 1
        break;
    end
    on = true;
    answer=false;
    while(answer == false)
        name = input('\nGive a name for file:\n>>', 's');
        if (isempty(strfind(name, '*')) == 0 || isempty(strfind(name, '<')) == 0 || isempty(strfind(name, '>')) == 0 ||
isempty(strfind(name, '?')) == 0 || isempty(strfind(name, '[')) == 0 || isempty(strfind(name, ']')) == 0 ||
isempty(strfind(name, ':')) == 0 || isempty(strfind(name, '|')) == 0 || isempty(strfind(name, '/')) == 0 ||
isempty(strfind(name, '"')) == 0 || isempty(strfind(name, '\')) == 0)
            cprintf('err', 'Filename must not contain characters: < > ? [ ] : | / " or *\n');
        else
            answer=true;
        end
    end
    if strcmpi('',name) == 1
        file = 'default';
        name = strcat(file, number);
    end
    controlDone = false;
    experimentDone = false;
    while (on == true)

        fprintf('This is the experiment.\n');

        nstage=10;
        xplot=1:0.1:9;     %scale for graphics-window

        %Initializing variables
        optx=zeros(2,1);
        betas=zeros(2,1);
        betacov=zeros(2,2,1);
        thetas=zeros(2,1);
        thetacov=zeros(2,2,1);
        theta_cf=zeros(2,1);
        lambda_cf=zeros(2,1);
        ismles=-ones(1);
        totaln=zeros(1);
        optx_c=zeros(2,1);
        beta_c=zeros(2,1);
        beta_c_found = 0;
        repeat = 42; %amount of repeats and also amount of repeats to add
        add = (round((repeat*6*2)/maxlimit))-repeat;

        fprintf('\nPressing any key will start the program by opening two windows,');
        fprintf('\none for answering and one for graph.');
        fprintf('\nPress any key to start and open windows...\n');
        pause;

        %Tells how many times one sound needs to be played.
        soundsLeft = repmat(repeat, maxlimit, 1);

        n_inc = 3;         %number of times we want to measure at the same point during one estimation
        figure(2);         %figure for graph
```

```
figure(1);        %figure for answering

%Draws red/green symbol for answering.
fill(a,b,'r')
axis square

fprintf('\nYou can move these two windows by dragging them from their frame.');
cprintf('green','\nNOTE: Clicking the window itself will start the actual program!');
fprintf('\nPress any key to continue test...\n') ;
pause();
pause(1);

%Calculates three points of interest (lower limit and two points near it)
%and plays corresponding sounds.
play_these_x = calculatePOI(minlimit, minlimit,maxlimit,maxlimit, n_inc);
[all_x all_y soundsLeft] = playSounds(play_these_x, soundsLeft, a, b, result);

%Calculates three points of interest (upper limit and two points near it)
%and plays corresponding sounds.
play_these_x = calculatePOI(maxlimit, minlimit,maxlimit,maxlimit, n_inc);
[all_x_inc all_y_inc soundsLeft] = playSounds(play_these_x, soundsLeft, a, b, result);

all_x = [all_x; all_x_inc];
all_y = [all_y; all_y_inc];
n=2;
ismle=0;
ismle_old=0;
notModelled = true;

notZero = true;
rounds = 0;
soundRunOut = {'No'};

if  (mean(all_y(all_x==minlimit)) < mean(all_y(all_x==maxlimit)))
   [optx_c beta_c betacov_c ismle]=optdesign(all_x,all_y);
   beta_c_found = beta_c;
   drawFigure(xplot, all_x, all_y, beta_c, beta_c_found,ismle, maxlimit);

   %Calculates the optimal design and likelihood estimates
   for i=1:nstage,

      if n>2
         [optx_c beta_c betacov_c ismle]=optdesign(all_x,all_y,beta_c);
      end
      optx(:,i)=optx_c';
      betas(:,i)=beta_c';
      betacov(:,:,i)=betacov_c;
      ismles(:,i)=ismle;
      if ismle==0,
         optx_c=min(maxlimit,max(minlimit,optx_c));
         optx_c = round(optx_c);
         [zeromax maxind]=max(all_x(all_y==0,1));
         [onemin minind]=min(all_x(all_y==1,1));

         play_these_x = calculatePOI(optx_c(1), zeromax, onemin, maxlimit, n_inc);
         [all_x_inc all_y_inc soundsLeft]=playSounds(play_these_x, soundsLeft, a, b, result);
      else
         [thetas(:,i),thetacov(:,:,i),theta_cf(:,i),lambda_cf(:,i)]=thetaest(beta_c,betacov_c);
         fprintf('\nStage: %d, Total n: %d\n',i,n)
```

```
                fprintf('Point(s) of measurement: %g %g\n',optx_c(1),optx_c(2))
                fprintf('Maximum likelihood estimates found!\n')
                fprintf('Parameter  Estimate       std\n')
                fprintf('a          %1.5f      %1.5f\n',betas(1,i),sqrt(betacov(1,1,i)))
                fprintf('b          %1.4f      %1.5f\n',betas(2,i),sqrt(betacov(2,2,i)))
                fprintf('theta      %1.3f      %1.5f\n',thetas(1,i),sqrt(thetacov(1,1,i)))
                fprintf('lambda     %1.4f      %1.5f\n',thetas(2,i),sqrt(thetacov(2,2,i)))
                rounds = i;

                %Saves the results to an excel-sheet called "Statistics"
                array = {'a'; 'b'; 'theta'; 'lambda'};
                cArray = cellstr(array);
                filename = name;
                xlswrite(filename, {'Parameter'}, 'Statistics', 'A1');
                xlswrite(filename, cArray, 'Statistics', 'A2');
                xlswrite(filename, {'Estimate'}, 'Statistics', 'B1');
                xlswrite(filename, [betas(1,i); betas(2,i); thetas(1,i); thetas(2,i)], 'Statistics', 'B2');
                xlswrite(filename, {'Standard deviation'}, 'Statistics', 'C1');
                xlswrite(filename, [sqrt(betacov(1,1,i)); sqrt(betacov(2,2,i)); sqrt(thetacov(1,1,i)); sqrt(thetacov(2,2,i))],
'Statistics', 'C2');
                notModelled = false;
                break;
            end
            beta_c_found = beta_c;
            drawFigure(xplot, all_x, all_y, beta_c, beta_c_found,ismle, maxlimit);

            ismle_old=ismle;
            n=n+n_inc;
            all_x = [all_x;all_x_inc];
            all_y = [all_y;all_y_inc];

            for j=1:9,
              if (soundsLeft(j) == 0)
                notZero = false;
              end
            end
            if notZero == false
              soundRunOut = {'Yes'};
              rounds = i;
              break;
            end
          end
        end

        %Points found by optimal design
        x_points=[];
        for i=minlimit:maxlimit
          if (mean(all_y(all_x==i)) < 1 && mean(all_y(all_x==i)) > 0)
            x_points = [x_points;i];
          end
        end

        %Three points that interest the most
        interest_x_points=[];

        max0 = 0;
        min1 = 0;

        %0 points found
```

```
if (isempty(x_points) == 1)
  for i=minlimit:maxlimit,
    if (mean(all_y(all_x==i)) == 0)
      max0 = i;
    end
  end
  j = maxlimit;
  while (j >= minlimit)
    if (mean(all_y(all_x==j)) == 1)
      min1 = j;
    end
    j = j-1;
  end
  x1 = max0;
  x2 = min1;
  if (x1 == maxlimit) || (x2 == minlimit)
    for i=1:maxlimit
      soundsLeft(i) = soundsLeft(i) + add;
    end
  elseif ((abs(maxlimit-x2))>(abs(minlimit-x1)))
    x3 = x2 + 1;
    interest_x_points = [x1;x2;x3];
  else
    x3 = x1 - 1;
    interest_x_points = [x3;x1;x2];
  end
  %1 point found
elseif (length(x_points) == 1)
  %only point at value x=1
  if (x_points(1) == minlimit)
    x1 = x_points(1);
    x2 = x_points(1) + 1;
    x3 = x_points(1) + 2;
    interest_x_points = [x1;x2;x3];
    %only point at value x=9
  elseif (x_points(1) == maxlimit)
    x1 = x_points(1);
    x2 = x_points(1) - 1;
    x3 = x_points(1) - 2;
    interest_x_points = [x3;x2;x1];
    %only point at value x=2...8
  else
    x1 = x_points(1);
    x2 = x_points(1) - 1;
    x3 = x_points(1) + 1;
    interest_x_points = [x2;x1;x3];
  end
  %2 points found
elseif (length(x_points) == 2)
  %points side by side
  if (x_points(1) == x_points(2) - 1)
    x1 = x_points(1);
    x2 = x_points(2);
    if (abs(maxlimit-x_points(2))>(abs(minlimit-x_points(1))))
      x3 = x_points(2) + 1;
      interest_x_points = [x1;x2;x3];
    else
      x3 = x_points(1) - 1;
      interest_x_points = [x3;x1;x2];
```

```
    end

    %1 value between two points
  elseif (x_points(1) == x_points(2) - 2)
    x1 = x_points(1);
    x2 = x_points(2);
    x3 = x_points(1) + 1;
    interest_x_points = [x1;x3;x2];
    %2 or more values between two points
  else
    midpoint = round((maxlimit/2));
    if (abs(x_points(1)-midpoint)) < (abs(x_points(2)-midpoint))
      x1 = x_points(1);
      if (x1 < midpoint)
        x2 = x1 + 1;
        x3 = x1 + 2;
        interest_x_points = [x1;x2;x3];
      elseif (x1 == midpoint)
        x2 = x1 - 1;
        x3 = x1 + 1;
        interest_x_points = [x2;x1;x3];
      else
        x2 = x1 - 1;
        x3 = x1 - 2;
        interest_x_points = [x3;x2;x1];
      end
    elseif (abs(x_points(1)-midpoint)) > (abs(x_points(2)-midpoint))
      x1 = x_points(2);
      if (x1 < midpoint)
        x2 = x1 + 1;
        x3 = x1 + 2;
        interest_x_points = [x1;x2;x3];
      elseif (x1 == midpoint)
        x2 = x1 - 1;
        x3 = x1 + 1;
        interest_x_points = [x2;x1;x3];
      else
        x2 = x1 - 1;
        x3 = x1 - 2;
        interest_x_points = [x3;x2;x1];
      end
    else
      x1 = midpoint - 1;
      x2 = midpoint;
      x3 = midpoint + 1;
      interest_x_points = [x1;x2;x3];
    end
  end
  %3 points found
  %3 points are side by side
elseif (length(x_points) == 3)
  if (x_points(1) + 2 == x_points(3))
    interest_x_points = [x_points(1); x_points(2); x_points(3)];
  else
    x2 = x_points(2);
    x1 = x2 - 1;
    x3 = x2 + 1;
    interest_x_points = [x1;x2;x3];
  end
```

```
    %4 points found
  elseif (length(x_points) == 4)
    x2 = round((x_points(2)+x_points(3))/2);
    x1 = x2 - 1;
    x3 = x2 + 1;
    interest_x_points = [x1;x2;x3];
    %5 ore more points found, interpreted as all points being equally
    %interesting
  else
    for i=1:maxlimit
      soundsLeft(i) = soundsLeft(i) + add;
    end
  end

  if (isempty(interest_x_points) == 0)
    for i=1:length(interest_x_points)
      soundsLeft(interest_x_points(i)) = soundsLeft(interest_x_points(i)) + repeat;
    end
  end
  %Plays rest of the sounds
  play_these_x = randomizeSoundsLeft(soundsLeft);

  try
    for i = 1 : size(play_these_x)
      [all_x_inc all_y_inc soundsLeft] = playSounds(play_these_x(i), soundsLeft, a, b, result);      %soita
randomääni, ja palauta x:n ja y:n arvo
      all_x = [all_x;all_x_inc];
      all_y = [all_y;all_y_inc];

      %pause so that test subject may rest a while in the middle of the test
      if (i == 230)
        cprintf('green','\nPause, press any key to continue!\n\n');
        pause();
      end

      if (mean(all_y(all_x==minlimit))<mean(all_y(all_x==maxlimit)))
        if (notModelled == false)
          if (beta_c_found == 0)
            [thetas(:,i),thetacov(:,:,i),theta_cf(:,i),lambda_cf(:,i)]=thetaest(beta_c,betacov_c);
            betas(:,i)=beta_c';
            betacov(:,:,i)=betacov_c;
            [optx_c beta_c betacov_c ismle]=optdesign(all_x,all_y);
            beta_c_found = beta_c;
            drawFigure(xplot, all_x, all_y, beta_c, beta_c_found,ismle, maxlimit);
          else
            try
              [thetas(:,i),thetacov(:,:,i),theta_cf(:,i),lambda_cf(:,i)]=thetaest(beta_c,betacov_c);
              betas(:,i)=beta_c';
              betacov(:,:,i)=betacov_c;
              [optx_c beta_c betacov_c ismle]=optdesign(all_x,all_y,beta_c);
              drawFigure(xplot, all_x, all_y, beta_c, beta_c_found,ismle, maxlimit);
            catch e
              cprintf('err','Error occured. Stepping out of loop...');
            end
          end
        end
      end

    end
```

```
        catch eee
          cprintf('err','Error in playing sounds, stopping playing and continuing to save...');
        end
        try
          cprintf('green','Experiment ends here!\n\n');
          %Saves the results to an excel-sheet called "Experiment"
          filename = name;
          xlswrite(filename, {'Sound'}, 'Experiment', 'A1');
          xlswrite(filename, all_x, 'Experiment', 'A2');
          xlswrite(filename, {'Response'}, 'Experiment', 'B1');
          xlswrite(filename, all_y, 'Experiment', 'B2');
          xlswrite(filename, {'Points of interest'}, 'Other', 'A1');
          if isempty(interest_x_points) == 1
            xlswrite(filename, {'None'}, 'Other', 'A2');
          else
            xlswrite(filename, interest_x_points, 'Other', 'A2');
          end
          xlswrite(filename, {'Design rounds'}, 'Other', 'B1');
          xlswrite(filename, rounds, 'Other', 'B2');
          xlswrite(filename, {'Did sound run out'}, 'Other', 'C1');
          xlswrite(filename, soundRunOut, 'Other', 'C2');
          xlswrite(filename, {'POI by model'}, 'Other', 'D1');
          if isempty(x_points) == 1
            xlswrite(filename, {'None'}, 'Other', 'D2');
          else
            xlswrite(filename, x_points, 'Other', 'D2');
          end
          if (notModelled == true)
            array = {'a'; 'b'; 'theta'; 'lambda'};
            cArray = cellstr(array);
            xlswrite(filename, {'Parameter'}, 'Statistics', 'A1');
            xlswrite(filename, cArray, 'Statistics', 'A2');
            xlswrite(filename, {'Estimate'}, 'Statistics', 'B1');
            xlswrite(filename, {'Standard deviation'}, 'Statistics', 'C1');
          end
          if (notModelled == false)
            try
              if (mean(all_y(all_x==minlimit))<mean(all_y(all_x==maxlimit)))
                xlswrite(filename, {'Final Estimate'}, 'Statistics', 'D1');
                xlswrite(filename, [betas(1,i); betas(2,i); thetas(1,i); thetas(2,i)], 'Statistics', 'D2');
                xlswrite(filename, {'Final Standard deviation'}, 'Statistics', 'E1');
                xlswrite(filename, [sqrt(betacov(1,1,i)); sqrt(betacov(2,2,i)); sqrt(thetacov(1,1,i));
sqrt(thetacov(2,2,i))], 'Statistics', 'E2');
              end
            catch ee
              cprintf('err','Error! Could not save the results.');
            end
          end
        catch esave
          cprintf('err','Error occured');
        end
        %Saves the figure of curves
        figure(2);
        filename1 = strcat(name, '_plot');
        saveas(gcf, filename1, 'png');

        answer = false;
        while (answer == false)
```

```
        result = input('Do you want to continue to the control treatment? Write y/n (yes/no) and press enter:\n>>',
's');
        switch result
          case 'y'
            result = 'c';
            answer = true;
          case 'n'
            fprintf('End of test!\n');
            fprintf('Press any key to quit.\n');
            answer = true;
            on = false;
            pause;
            error('');
            close all;
          otherwise
            cprintf('err','INCORRECT INPUT');
            cprintf('text', ', please try again.\n\n');
        end
      end

      %Control treatment starts
      if strcmpi('c',result) == 1
        fprintf('This is the control treatment, press any key to begin.\n');
        pause();

        all_x = xlsread(filename, 'Experiment', 'A2:A600');
        try
          playThese(all_x);
        catch ec
          cprintf('err','Unable to play sound. Saving the results...');
        end
        %Saves the results to an excel-sheet called "Control"
        %filename = strcat(name, 'control');
        filename = name;
        xlswrite(filename, {'Sound'}, 'Control', 'A1');
        xlswrite(filename, all_x, 'Control', 'A2');

        fprintf('End of the test!\n');
        fprintf('Press any key to quit.\n');
        pause;
        on = false;
        error('');
        close all;
      end
    end
```

Liite 5: Aliohjelmien ohjelmakoodit

calculatePOI-aliohjelma:

```
function [ play_these_x ] = calculatePOI( x, a, b, numberOfVoices, n_inc)
%Calculates three points of interest and randomizes them so that there
%isn't same values of x in a row. All the points are round to integers.
%
%Input:
%x              The point of x we are interested
%a              Maximum value of x where response is 0 (lower limit for x)
%b              Minimum value of x where response is 1 (upper limit for x)
%numberOfVoices    The upper limit for x
%n_inc           The number of times we want to measure at the
%               same point during one estimation
%
%Output:
%play_these_x      Vector of points to be played
%
%Author: Minna Lehtomäki
%Version 1.0 2013-23-03
%
  interval = abs(b-a);
  quarter = interval/4;

  POI_a = round(x-quarter);
  POI_b = round(x+quarter);

  if POI_a == x
    POI_a = x-1;
  end

  if POI_b == x
    POI_b = x+1;
  end

  if x == 1
    POI_a = x + 2 * (POI_b-x);
  end

  if x == numberOfVoices
    POI_b = x - 2 * (x - POI_a);
  end

  for i=1:n_inc
      play_these_x(1 + 3*(i-1)) = POI_a;
      play_these_x(2 + 3*(i-1)) = POI_b;
      play_these_x(3 + 3*(i-1)) = x;
  end

  randIsOk = false;
  %Randomizes the values of x
  while randIsOk == false
    play_these_x= play_these_x(randperm(length(play_these_x)));

    randIsOk = true;
    for j = 1 : n_inc * 3 -1
```

```
        if play_these_x(j) == play_these_x(j+1)
          randIsOk = false;
          break;
        end
      end

  end
  play_these_x = play_these_x';
end
```

drawFigure-aliohjelma:

```
function [ ] = drawFigure(xplot, xpoint, ypoint, beta_c, beta_c_found, beta_cIsFound, numberOfSounds)
%Draws a graph of the measured data using logistic link.
%
%Input:
%
%xplot            Scale for x-axis
%xpoint           Vector of measured points x
%ypoint           Vector of measured responses Y
%beta_c           Estimated values of parameters a and b
%beta_c_found     Maximum likelihood estimates of parameters a and b
%beta_cIsFound    Are the maximum likelihood estimates found, 0=No
%numberOfSounds   Upper limit for x
%
%Uses:            logistic_cdf.m
%
%Author: Minna Lehtomäki
%Version 1.0 2013-22-02

  %Initializing variables
  data=[];
  x = [];
  y = [];
  Empties = 0;
  for i = 1: numberOfSounds
    Empty = true;
    index = 1;
    for j = 1 : size(xpoint)
      if xpoint(j) == i
        data(i - Empties, index) = ypoint(j);
        index = index + 1;
        Empty= false;
      end
    end
    if Empty == true
      Empties = Empties + 1;
    else
      x(length(x) + 1) = i;
      y_sum = 0;
      for k=1:index - 1
        y_sum = y_sum + data(i - Empties, k);
      end
      y(length(y) +1) = y_sum / (index - 1);
    end
  end

  figure(2);
```

```
  lwidth=1.1;
  msize=8;
  fsize=11;
  plottitle='Response curve';
  clf
  if (beta_cIsFound == 0)
    %Graph for estimated response curve
    plot(xplot,logistic_cdf(beta_c,xplot),'g',...
    x,y,'kx','MarkerSize',msize)
    legend('Estimated','Measured point',2)
  elseif (beta_c_found == beta_c)
    plot(xplot,logistic_cdf(beta_c,xplot),'g',...
    x,y,'kx','MarkerSize',msize)
    legend('Final','Measured point',2)
  else
    %Response curves of all data and of data gathered by optdesign.m
    plot(xplot,logistic_cdf(beta_c,xplot),'g',...
    xplot,logistic_cdf(beta_c_found,xplot),'b',...
    x,y,'kx','MarkerSize',msize)
    legend('Final','Estimated','Measured point',2)
  end
  hold on
  title(plottitle,'FontSize',fsize)
  legend('boxoff');
  set(gca, 'Ylim',[0,1]);
  set(gca, 'Xlim',[1,9]);
  xlabel('Stimulus from ba to pa','FontSize',fsize);
  ylabel('Percent identification as pa','FontSize',fsize,'Rotation',90);
  set(gca,'LineWidth',lwidth);
  set(gca,'FontSize',fsize);
  h=findobj('Type','line');
  set(h,'LineWidth',lwidth);

end
```

logistic_cdf-aliohjelma:

```
function p=logistic_cdf(beta_,x)
%Calculates the cumulative distribution function of the logistic distribution
%p=logistic_cdf(beta_,x)
%
%Input:
%beta_(1)   scale parameter of the logistic distribution = a
%beta_(2)   location parameter of the logistic distribution = b
%x         vector of points where the value of the cumulative
%          distribution function is calculated.
%Output:
%p          vector of the values of the cumulative distribution
%          function in points x.
%
a=beta_(1);
b=beta_(2);
p=1./(1+exp(-(a*x+b)));
```

logistic_cov-aliohjelma:

```
function lcov=logistic_cov(beta_,x,y)
%Calculates the covariance matrix for parameters a and b in logistic regression
%lcov=logistic_cov(beta_,x,y)
%
%Input:
%beta_(1)   scale parameter of the logistic distribution = a
%beta_(2)   location parameter of the logistic distribution = b
%x         vector of the covariate points
%y         vector of the response values
%
%Output:
%lcov      The (asymptotic) covariance matrix for beta_
%

a=beta_(1);
b=beta_(2);
z=a*x+b;
expz=exp(z);
%In order to avoid numerical problems, very small values are truncated to
%zero.
zind=((z>-10) & (z<5));
expz_=expz(zind);
d2=zeros(size(z));
d2(zind)=(-expz_+exp(z(zind)+expz_)-exp(2*z(zind)+expz_))./(-1+exp(expz_)).^2;
%The covariance matrix is calculated as the inverse of the observed Fisher
%information.
J(1,1)=-sum(y.*(x.^2).*d2-(1-y).*(x.^2).*expz);
J(1,2)=-sum(y.*x.*d2-(1-y).*x.*expz);
J(2,1)=J(1,2);
J(2,2)=-sum(y.*d2-(1-y).*expz);
lcov=inv(J);
```

logistic_logl-aliohjelma:

```
function logl=logistic_logl(beta_,x,y)
%Calculates the log-likehood for logistic regression
%logl=logistic_logl(beta_,x,y)
%
%Input:
%beta_(1)   scale parameter of the logistic distribution = a
%beta_(2)   location parameter of the logistic distribution = b
%x         vector of the covariate points
%y         vector of the response values
%
%Output:
%logl      The value of the log-likelihood multiplied by -1
%

Fx=max(0.00000001,min(0.99999999,logistic_cdf(beta_,x)));
logl=sum(y.'*log(Fx)+(1-y).'*log(1-Fx));
logl=-logl;
```

optdesign-aliohjelma (Karvanen, 2008, Optdesign-package):

```
function [optx,betaest,betacov,IsMLE]=optdesign(xx,yy,beta0,transform)
%[optx,betaest,betacov]=optdesign(xx,yy)  - Calculates the optimal design
%for binary response data
%
%Version 1.2 2013-03-26 by Minna Lehtomäki
%Version 1.1 2006-05-06
%Version 1.0 2005-10-01
%Copyright Juha Karvanen 2005
%The program is distributed under the terms of the GNU General Public License
%For details of the copyright see the files readme.txt and gpl.txt provided in the same package.
%
%The program calculates the optimal design for binary response data. More
%generally, the program calculates the optimal covariate points for binary response
%with logit link. The optimality is defined as D-optimality and parameters are estimated
%by maximum likelihood estimation. The data is assumed to follow
%logistic distribution
%
%           P(Y=1) = 1/(1+exp(-(a*x+b)),
%
%where a and b are the model parameters, x is the covariate variable and Y is the binary
%response. All data collected so far is given as an input to the program which then returns
%the optimal covariate points to be measured in next stage.
%
%Sequential design with optdesign.m:
%
%           1) Obtain the limits of the initial interval: xmin and xmax
%           2) Measure at points xmin and xmax. Let x contain the points
%              of measurement and y contain the measured responses.
%              (Sometimes it can be directly assumed that the response for
%               xmin is 0 and the response for xmax is 1.)
%           3) Call opdesign.m
%              [optx,betaest,betacov,IsMLE]=optdesign(x,y)
%           4) If the required accuracy of betaest has been achieved, stop the
%              procedure; otherwise proceed to the next step.
%           5) Measure at the points given by optx. Append the points of measurement
%              to x. Append the measured responses to y.
%           6) Go to step 3.
%
%The function was originally intended to be used for sequential designs of switching
%measurements but it was modified to use logistic distribution instead of Gompertz
%distribution.
%
%
%Reference to original paper:
%                                   Juha Karvanen, Juha J. Vartiainen, Andrey Timofeev and Jukka
%           Pekola, Experimental Designs for Binary Data in Switching
%           Measurements on Superconducting Josephson Junctions, 2005,
%           http://ltl.tkk.fi/PICO/optdesign/
%
%Uses:        logistic_logl.m
%           logistic_cov.m
%           logistic_cdf (through logistic_logl.m)
%
%Input:
%xx          Points measured so far,
%              size(xx)=[n 1], where n is the total number of points
```

```
%               measured so far.
%yy          Measured binary response for points xx, 0 or 1, size(yy)=[n 1].
%beta0        (optional) initial values of a and b for maximum likelihood
%               estimation, beta0=[a0 b0], lenght(beta0)=2.
%transform                       (Not in use in this version)
%
%Output:
%optx                            Points to be measured at the next stage, length(optx)=2
%               If IsMLE==1 measure at both optx(1) and optx(2)
%               If IsMLE==0 measure only at optx(1).
%betaest                 Estimated values of parameters a and b
%               If IsMLE==0, an ad-hoc estimate is returned.
%betacov       Estimated covariance matrix for parameters a and b,
%               size(betacov)=[2,2]
%               If IsMLE==0, betacov cannot be calculated and a matrix of NaN:s is returned.
%IsMLE         Are the estimates maximum likelihood estimates? 0 or 1.
%


% Checking input %
if nargin<2
   error('Input variables xx and yy are required.')
end
[nx px]=size(xx);
[ny py]=size(yy);
if px~=1 | py~=1 | nx~=ny
   error('Input variables xx and yy should be column vectors of same size: size(xx)==size(yy)==[n 1]')
end
if sum(yy~=0 & yy~=1)>0
   error('Input variable yy may contain only values 0 and 1.')
end
if not(isreal(xx))
   error('Input variable xx must be real valued.')
end
if (var(xx)==0)
   error('Input variable xx must have at least two distinct values.')
end
if nargin>=3 %Checking for beta0
   if not(isreal(beta0)) | length(beta0)~=2
      error('Input variable beta0 must be a real valued vector of two values [a0>0 b0].')
   end
   if beta0(1)<=0
      error('Input variable beta0 must be a real valued vector of two values [a0>0 b0].')
   end
end

% MATLAB version
verz=version;
% Initialization
betaest=[NaN; NaN];
betacov=[NaN NaN; NaN NaN];
IsMLE=0;
epsilon=1;

[minx minxind]=min(xx);
[maxx maxxind]=max(xx);
if (mean(yy(xx==minx))>=mean(yy(xx==maxx)))
   warning('Assumption a>0 violated. Check the data.')
end
```

```
%Checking the existence of maximum likelihood estimate
[zeromax maxind]=max(xx(yy==0,1));
[onemin minind]=min(xx(yy==1,1));
if zeromax<onemin,
   x1=(zeromax+onemin)/2;
   optx=[x1 x1];
   %When MLE does not exist, an ad-hoc estimate is returned.
   a=9.6/(maxx-minx); %9.6=-log(-log(0.9995))+log(-log(0.0005))
   b=log(-log(0.5))-a*(zeromax+onemin)/2;
            betaest=[a b];
elseif onemin==zeromax,
   %If onemin same as zeromax, one of the points needed is found and the other is searched in the neighbourhood
   epsilonsign=sign(0.5-mean(yy));
   if (epsilonsign==0)
      epsilonsign=2*(rand(1)-0.5)-1; %randomly 1 or -1
   end
   if (epsilonsign==1)
      epsilon=(min(xx(xx>zeromax | xx==maxx))-zeromax)/2;
   end
   if (epsilonsign==-1)
      epsilon=(max(xx(xx<zeromax | xx==minx))-zeromax)/2;
   end
   x1=zeromax+epsilon;
   optx=[x1 x1];

   %When MLE does not exist, an ad-hoc estimate is returned.
   a=9.6/(maxx-minx); %9.6=-log(-log(0.9995))+log(-log(0.0005))
   b=log(-log(0.5))-a*(zeromax+onemin)/2;
            betaest=[a b];
else
   IsMLE=1;
   if nargin==2 %No beta0 is given as input, an ad-hoc beta0 is calculated.
      a=9.6/(maxx-minx); %9.6=-log(-log(0.9995))+log(-log(0.0005))
      b=log(-log(0.5))-a*(zeromax+onemin)/2;
                  beta0=[a b];
            end
            x=xx;
            y=yy;
   %The optimal points (canonical form)
            z2=1.5434;
   z1=-1.5434;

   %The name of optimization procedure is different in the earlier
   %MATLAB versions. We use here the Nelder-Mead optimization that is slower
   %than the gradient based methods but avoids the risk of the numerical
   %instability of gradient. In our application (switching measurements)
   %the stability is critical.
   if str2num(verz(1))<=5,
      fminopt=FOPTIONS;
      fminopt(14)=4000;
      betaest=fmins('logistic_logl',beta0,fminopt,[],x,y);
   else
      fminopt=optimset('MaxIter',4000);
      betaest=fminsearch('logistic_logl',beta0,fminopt,x,y);
   end
            a=betaest(1);
            b=betaest(2);
            x1=(z1-b)/a;
            x2=(z2-b)/a;
```

```
            optx=[x1 x2];
  betacov=logistic_cov(betaest,x,y);
end
```

playSounds-aliohjelma:

```
function [all_x,all_y, soundsLeft] = playSounds(x, soundsLeft, a, b, result)
%Plays sounds, sends triggers to EEG-machine, controls answering window and
%reads response input.
%
%Input:
%x           Vector of points which sounds we want to play
%soundsLeft     How many times one sound needs to be played
%a           Parameter for answering windows's symbol
%b           Parameter for answering windows's symbol
%result       Is this the control treatment (c) or the experiment (e)
%
%Output:
%all_x       Vector of all measured points x
%all_y       Vector of all responses Y
%soundsLeft     How many times one sound needs to be played
%
%Author: Minna Lehtomäki
%Version 1.0 2013-20-02

  %Read sounds
  [y1,Fs1] = wavread('aanet\5ms_voicingbar.wav');
  [y2,Fs2] = wavread('aanet\0msVOT.wav');
  [y3,Fs3] = wavread('aanet\9msVOT.wav');
  [y4,Fs4] = wavread('aanet\17msVOT.wav');
  [y5,Fs5] = wavread('aanet\25msVOT.wav');
  [y6,Fs6] = wavread('aanet\34msVOT.wav');
  [y7,Fs7] = wavread('aanet\44msVOT.wav');
  [y8,Fs8] = wavread('aanet\57msVOT.wav');
  [y9,Fs9] = wavread('aanet\64msVOT.wav');

  paraport = digitalio('parallel','LPT1');
  line1=addline(paraport,0:3,'out');
  trg0=[0 0 0 0];
  trg1=[1 0 0 0];
  trg2=[0 1 0 0];
  trg3=[1 1 0 0];
  trg4=[0 0 1 0];
  trg5=[1 0 1 0];
  trg6=[0 1 1 0];
  trg7=[1 1 1 0];
  trg8=[0 0 0 1];
  trg9=[1 0 0 1];
 putvalue(paraport, trg0);

  for i=1:size(x),
    if (x(i) == 1)
      putvalue(paraport, trg1);
      soundsc(y1, Fs1);
      all_x(i) = 1;
      soundsLeft(1) = soundsLeft(1) - 1;
    end
    if (x(i) == 2)
```

```
   putvalue(paraport, trg2);
   soundsc(y2, Fs2);
   all_x(i) = 2;
   soundsLeft(2) = soundsLeft(2) - 1;
end
if (x(i) == 3)
   putvalue(paraport, trg3);
   soundsc(y3, Fs3);
   all_x(i) = 3;
   soundsLeft(3) = soundsLeft(3) - 1;
end
if (x(i) == 4)
   putvalue(paraport, trg4);
   soundsc(y4, Fs4);
   all_x(i) = 4;
   soundsLeft(4) = soundsLeft(4) - 1;
end
if (x(i) == 5)
   putvalue(paraport, trg5);
   soundsc(y5, Fs5);
   all_x(i) = 5;
   soundsLeft(5) = soundsLeft(5) - 1;
end
if (x(i) == 6)
   putvalue(paraport, trg6);
   soundsc(y6, Fs6);
   all_x(i) = 6;
   soundsLeft(6) = soundsLeft(6) - 1;
end
if (x(i) == 7)
   putvalue(paraport, trg7);
   soundsc(y7, Fs7);
   all_x(i) = 7;
   soundsLeft(7) = soundsLeft(7) - 1;
end
if (x(i) == 8)
   putvalue(paraport, trg8);
   soundsc(y8, Fs8);
   all_x(i) = 8;
   soundsLeft(8) = soundsLeft(8) - 1;
end
if (x(i) == 9)
   putvalue(paraport, trg9);
   soundsc(y9, Fs9);
   all_x(i) = 9;
   soundsLeft(9) = soundsLeft(9) - 1;
end

%Answering window
pause(1);
putvalue(paraport, trg0);
figure(1);              %calls right figure active
clf;                    %clears figure
fill(a,b,'g')           %fills symbol with green
waitforbuttonpress;
figure(1);
clf;
fill(a,b,'r')           %fills symbol with red
pause(1);
```

```
      %Which mouse button was pressed
      mouseside=get(gcf,'SelectionType');
      if strcmp('normal', mouseside);
         all_y(i) = 0;
      else %if strcmp('alt', mouseside);
         all_y(i) = 1;
      end
   end

   all_x = all_x';
   all_y = all_y';
end
```

playThese-aliohjelma:

```
function [] = playThese(x)
%Plays sounds, sends triggers to EEG-machine, controls answering window and
%reads response input.
%
%Input:
%x          Vector of points which sounds we want to play
%
%
%Author: Minna Lehtomäki
%Version 1.0 2013-20-02

%Read sounds
[y1,Fs1] = wavread('aanet\5ms_voicingbar.wav');
[y2,Fs2] = wavread('aanet\0msVOT.wav');
[y3,Fs3] = wavread('aanet\9msVOT.wav');
[y4,Fs4] = wavread('aanet\17msVOT.wav');
[y5,Fs5] = wavread('aanet\25msVOT.wav');
[y6,Fs6] = wavread('aanet\34msVOT.wav');
[y7,Fs7] = wavread('aanet\44msVOT.wav');
[y8,Fs8] = wavread('aanet\57msVOT.wav');
[y9,Fs9] = wavread('aanet\64msVOT.wav');

paraport = digitalio('parallel','LPT1');
line1=addline(paraport,0:3,'out');
trg0=[0 0 0 0];
trg1=[1 0 0 0];
trg2=[0 1 0 0];
trg3=[1 1 0 0];
trg4=[0 0 1 0];
trg5=[1 0 1 0];
trg6=[0 1 1 0];
trg7=[1 1 1 0];
trg8=[0 0 0 1];
trg9=[1 0 0 1];
putvalue(paraport, trg0);

for i=1:size(x),
   if (x(i) == 1)
      putvalue(paraport, trg1);
      soundsc(y1, Fs1);

   end
```

```
  if (x(i) == 2)
    putvalue(paraport, trg2);
    soundsc(y2, Fs2);

  end
  if (x(i) == 3)
    putvalue(paraport, trg3);
    soundsc(y3, Fs3);

  end
  if (x(i) == 4)
    putvalue(paraport, trg4);
    soundsc(y4, Fs4);

  end
  if (x(i) == 5)
    putvalue(paraport, trg5);
    soundsc(y5, Fs5);

  end
  if (x(i) == 6)
    putvalue(paraport, trg6);
    soundsc(y6, Fs6);

  end
  if (x(i) == 7)
    putvalue(paraport, trg7);
    soundsc(y7, Fs7);

  end
  if (x(i) == 8)
    putvalue(paraport, trg8);
    soundsc(y8, Fs8);

  end
  if (x(i) == 9)
    putvalue(paraport, trg9);
    soundsc(y9, Fs9);

  end
  pause(0.1);
  putvalue(paraport, trg0);
  pause(1);
end
```

randomizeSoundsLeft-aliohjelma:

```
function [ play_these_x ] = randomizeSoundsLeft( soundsLeft )
%Randomizes sounds that haven't been played yet.
%
%Input:
%soundsLeft     How many times one sound needs to be played
%
%Output:
%play_these_x   Randomized vector of sounds to be played
%
%Author: Minna Lehtomäki
%Version 1.0 2013-23-02
```

```
%Version 1.1 2014-15-05
%

  SoundsLeftCopy = soundsLeft;
  index = 1;

  ContinueLoop = true;
  %Arranges the remaining sounds into a vector randomly
  while ContinueLoop == true;
    IsGood = false;

    while IsGood == false
      x = round(1+ 8*rand(1));          %1...9 random sounds
      if SoundsLeftCopy(x) > 0
        IsGood = true;
        SoundsLeftCopy(x) = SoundsLeftCopy(x) - 1;
      end
    end

    play_these_x(index) = x;
    index = index + 1;


    ContinueLoop = false;
    %Checks whether all the sounds have been replayed enough
    for soundFile = 1: size(SoundsLeftCopy, 1)
      if (SoundsLeftCopy(soundFile) > 0)
        ContinueLoop = true;
      end
    end
  end

  %Transpose of matrix play_these_x
  play_these_x = play_these_x';

  IsGood = false;
  iterations = 1;
  %Randomizes sounds so that there is no more than 2 same sounds after
  %each other. Tries the randomization max 200 times before continues.
  while IsGood == false && iterations < 200
    play_these_x = play_these_x(randperm(length(play_these_x)));
    IsGood = true;
    for ii = 1 : size(play_these_x) - 2
      if  (play_these_x(ii) == play_these_x(ii+1))
        if play_these_x(ii) == play_these_x(ii+2)
          IsGood = false;
          iterations = iterations + 1;
          break;
        end
      end
    end
  end
end
```

thetaest-aliohjelma :

```
function [thetaest,thetacov,theta_cf,lambda_cf]=thetaest(betaest,betacov)
%Calculates estimates of theta and lambda, in logistic regression
%[thetaest,thetacov,theta_cf,lambda_cf]=thetaest(betaest,betacov)
%
%Here Theta stand for the 50% point of the response curve, while
%lambda is the difference of the 10% and 90% points, thus characterizing
%the width of the response curve.
%
%Input:
%       betaest         estimates of the parameters a and b
%       betacov         covariance matrix for the parameters a and b
%
%Output:
%       thetaest   estimates of parameters theta (the middle point) and lambda (the width of the curve)
%       thetacov   covariance matrix for the parameters theta and lambda
%       theta_cf   95% confidence interval for the parameter theta
%       lambda_cf       95% confidence interval for the parameter lambda
%
%Juha Karvanen 2005-05-09
%Modified for logistic regression: Minna Lehtomäki 2013-03-36
q1= 1./(1+exp(-(0.5)));
q2=(1./(1+exp(-(0.9))))-(1./(1+exp(-(0.1))));
a=betaest(1);
b=betaest(2);
theta=(q1-b)/a;
lambda=q2/a;
thetaest=[theta lambda];
A=[(b-q1)/(a^2) -1/a;-q2/(a^2) 0];
thetacov=A*betacov*A';
stds=sqrt(diag(thetacov));
%Normal approximation is used for the confidence intervals.
theta_cf=[theta-1.96*stds(1) theta+1.96*stds(1)];
lambda_cf=[lambda-1.96*stds(2) lambda+1.96*stds(2)];

thetaest=thetaest';
thetacov=thetacov';
theta_cf=theta_cf';
lambda_cf=lambda_cf';
```

Liite 6: draw-ohjelman MATLAB-koodi

```
%If program fails to draw the response curve, this code can be used to draw
%the picture and save it by providing the file name
%
%Uses:                  optdesign.m
%                                drawFigure.m
%                                cprintf.m
%
%Author: Minna Lehtomäki
%Version 1.0 2014-05-23
%

filename = 'lauril';
xplot=1:0.1:9;
maxlimit=9;
beta_c_found = 0;
rounds = xlsread(filename, 'Other', 'B2:B3');

try
   if (rounds==0)
      x = xlsread(filename, 'Experiment', 'A2:A19');
      y = xlsread(filename, 'Experiment', 'B2:B19');
      [optx_c beta_c betacov_c ismle]=optdesign(x,y);
      beta_c_found = beta_c;
   else
      rounds1 = 19+((rounds-1)*9);
      cs1 = ['A2:A', num2str(rounds1)];
      cs2 = ['B2:B', num2str(rounds1)];
      x = xlsread(filename, 'Experiment', cs1);
      y = xlsread(filename, 'Experiment', cs2);
      [optx_c beta_c betacov_c ismle]=optdesign(x,y);
      beta_c_found = beta_c;
   end
catch e1
   cprintf('green', 'Trying to continue...\n');
end

all_x = xlsread(filename, 'Experiment', 'A2:A600');
all_y = xlsread(filename, 'Experiment', 'B2:B600');
[optx_c beta_c betacov_c ismle]=optdesign(all_x,all_y);
if (beta_c_found == 0)
   beta_c_found = beta_c;
end

drawFigure(xplot, all_x, all_y, beta_c, beta_c_found, ismle, maxlimit);

filename1 = strcat(filename, '_plot2');
saveas(gcf, filename1, 'png');
```

Liite 7: cprintf-ohjelman MATLAB-koodi

Altman (2012), cprintf.

```
function count = cprintf(style,format,varargin)
% CPRINTF displays styled formatted text in the Command Window
%
% Syntax:
%   count = cprintf(style,format,...)
%
% Description:
%   CPRINTF processes the specified text using the exact same FORMAT
%   arguments accepted by the built-in SPRINTF and FPRINTF functions.
%
%   CPRINTF then displays the text in the Command Window using the
%   specified STYLE argument. The accepted styles are those used for
%   Matlab's syntax highlighting (see: File / Preferences / Colors /
%   M-file Syntax Highlighting Colors), and also user-defined colors.
%
%   The possible pre-defined STYLE names are:
%
%     'Text'               - default: black
%     'Keywords'           - default: blue
%     'Comments'           - default: green
%     'Strings'            - default: purple
%     'UnterminatedStrings'  - default: dark red
%     'SystemCommands'       - default: orange
%     'Errors'             - default: light red
%     'Hyperlinks'         - default: underlined blue
%
%     'Black','Cyan','Magenta','Blue','Green','Red','Yellow','White'
%
%   STYLE beginning with '-' or '_' will be underlined. For example:
%       '-Blue' is underlined blue, like 'Hyperlinks';
%       '_Comments' is underlined green etc.
%
%   STYLE beginning with '*' will be bold (R2011b+ only). For example:
%       '*Blue' is bold blue;
%       '*Comments' is bold green etc.
%   Note: Matlab does not currently support both bold and underline,
%       only one of them can be used in a single cprintf command. But of
%       course bold and underline can be mixed by using separate commands.
%
%   STYLE also accepts a regular Matlab RGB vector, that can be underlined
%   and bolded: -[0,1,1] means underlined cyan, '*[1,0,0]' is bold red.
%
%   STYLE is case-insensitive and accepts unique partial strings just
%   like handle property names.
%
%   CPRINTF by itself, without any input parameters, displays a demo
%
% Example:
%   cprintf;   % displays the demo
%   cprintf('text',   'regular black text');
%   cprintf('hyper',  'followed %s','by');
%   cprintf('key',    '%d colored', 4);
%   cprintf('-comment','& underlined');
```

```
%   cprintf('err',    'elements\n');
%   cprintf('cyan',   'cyan');
%   cprintf('_green', 'underlined green');
%   cprintf(-[1,0,1], 'underlined magenta');
%   cprintf([1,0.5,0],'and multi-\nline orange\n');
%   cprintf('*blue',  'and *bold* (R2011b+ only)\n');
%   cprintf('string');  % same as fprintf('string') and cprintf('text','string')
%
% Bugs and suggestions:
%   Please send to Yair Altman (altmany at gmail dot com)
%
% Warning:
%   This code heavily relies on undocumented and unsupported Matlab
%   functionality. It works on Matlab 7+, but use at your own risk!
%
%   A technical description of the implementation can be found at:
%   <a
href="http://undocumentedmatlab.com/blog/cprintf/">http://UndocumentedMatlab.com/blog/cprintf/</a>
%
% Limitations:
%   1. In R2011a and earlier, a single space char is inserted at the
%      beginning of each CPRINTF text segment (this is ok in R2011b+).
%
%   2. In R2011a and earlier, consecutive differently-colored multi-line
%      CPRINTFs sometimes display incorrectly on the bottom line.
%      As far as I could tell this is due to a Matlab bug. Examples:
%        >> cprintf('-str','under\nline'); cprintf('err','red\n'); % hidden 'red', unhidden '_'
%        >> cprintf('str','regu\nlar'); cprintf('err','red\n'); % underline red (not purple) 'lar'
%
%   3. Sometimes, non newline ('\n')-terminated segments display unstyled
%      (black) when the command prompt chevron ('>>') regains focus on the
%      continuation of that line (I can't pinpoint when this happens).
%      To fix this, simply newline-terminate all command-prompt messages.
%
%   4. In R2011b and later, the above errors appear to be fixed. However,
%      the last character of an underlined segment is not underlined for
%      some unknown reason (add an extra space character to make it look better)
%
%   5. In old Matlab versions (e.g., Matlab 7.1 R14), multi-line styles
%      only affect the first line. Single-line styles work as expected.
%      R14 also appends a single space after underlined segments.
%
%   6. Bold style is only supported on R2011b+, and cannot also be underlined.
%
% Change log:
%   2012-08-09: Graceful degradation support for deployed (compiled) and non-desktop applications; minor bug
fixes
%   2012-08-06: Fixes for R2012b; added bold style; accept RGB string (non-numeric) style
%   2011-11-27: Fixes for R2011b
%   2011-08-29: Fix by Danilo (FEX comment) for non-default text colors
%   2011-03-04: Performance improvement
%   2010-06-27: Fix for R2010a/b; fixed edge case reported by Sharron; CPRINTF with no args runs the demo
%   2009-09-28: Fixed edge-case problem reported by Swagat K
%   2009-05-28: corrected nargout behavior sugegsted by Andreas Gäb
%   2009-05-13: First version posted on <a
href="http://www.mathworks.com/matlabcentral/fileexchange/authors/27420">MathWorks File Exchange</a>
%
% See also:
%   sprintf, fprintf
```

```
% License to use and modify this code is granted freely to all interested, as long as the original author is
% referenced and attributed as such. The original author maintains the right to be solely associated with this work.

% Programmed and Copyright by Yair M. Altman: altmany(at)gmail.com
% $Revision: 1.08 $  $Date: 2012/10/17 21:41:09 $

 persistent majorVersion minorVersion
 if isempty(majorVersion)
    %v = version; if str2double(v(1:3)) <= 7.1
    %majorVersion = str2double(regexprep(version,'^(\d+).*','$1'));
    %minorVersion = str2double(regexprep(version,'^\d+\.(\d+).*','$1'));
    %[a,b,c,d,versionIdStrs]=regexp(version,'^(\d+)\.(\d+).*');  %#ok unused
    v = sscanf(version, '%d.', 2);
    majorVersion = v(1); %str2double(versionIdStrs{1}{1});
    minorVersion = v(2); %str2double(versionIdStrs{1}{2});
 end

 % The following is for debug use only:
 %global docElement txt el
 if ~exist('el','var') || isempty(el),  el=handle([]);  end  %#ok mlint short-circuit error ("used before defined")
 if nargin<1, showDemo(majorVersion,minorVersion); return;  end
 if isempty(style),  return;  end
 if all(ishandle(style)) && length(style)~=3
    dumpElement(style);
    return;
 end

 % Process the text string
 if nargin<2, format = style; style='text';  end
 %error(nargchk(2, inf, nargin, 'struct'));
 %str = sprintf(format,varargin{:});

 % In compiled mode
 try useDesktop = usejava('desktop'); catch, useDesktop = false; end
 if isdeployed | ~useDesktop %#ok<OR2> - for Matlab 6 compatibility
    % do not display any formatting - use simple fprintf()
    % See: http://undocumentedmatlab.com/blog/bold-color-text-in-the-command-window/#comment-103035
    % Also see: https://mail.google.com/mail/u/0/?ui=2&shva=1#all/1390a26e7ef4aa4d
    % Also see: https://mail.google.com/mail/u/0/?ui=2&shva=1#all/13a6ed3223333b21
    count1 = fprintf(format,varargin{:});
 else
    % Else (Matlab desktop mode)
    % Get the normalized style name and underlining flag
    [underlineFlag, boldFlag, style] = processStyleInfo(style);

    % Set hyperlinking, if so requested
    if underlineFlag
       format = ['<a href="">' format '</a>'];

       % Matlab 7.1 R14 (possibly a few newer versions as well?)
       % have a bug in rendering consecutive hyperlinks
       % This is fixed by appending a single non-linked space
       if majorVersion < 7 || (majorVersion==7 && minorVersion <= 1)
          format(end+1) = ' ';
       end
    end

    % Set bold, if requested and supported (R2011b+)
```

```
            if boldFlag
               if (majorVersion > 7 || minorVersion >= 13)
                  format = ['<strong>' format '</strong>'];
               else
                  boldFlag = 0;
               end
            end

            % Get the current CW position
            cmdWinDoc = com.mathworks.mde.cmdwin.CmdWinDocument.getInstance;
            lastPos = cmdWinDoc.getLength;

            % If not beginning of line
            bolFlag = 0;  %#ok
            %if docElement.getEndOffset - docElement.getStartOffset > 1
               % Display a hyperlink element in order to force element separation
               % (otherwise adjacent elements on the same line will be merged)
               if majorVersion<7 || (majorVersion==7 && minorVersion<13)
                  if ~underlineFlag
                     fprintf('<a href=""> </a>');  %fprintf('<a href=""> </a>\b');
                  elseif format(end)~=10  % if no newline at end
                     fprintf(' ');  %fprintf(' \b');
                  end
               end
               %drawnow;
               bolFlag = 1;
            %end

            % Get a handle to the Command Window component
            mde = com.mathworks.mde.desk.MLDesktop.getInstance;
            cw = mde.getClient('Command Window');
            xCmdWndView = cw.getComponent(0).getViewport.getComponent(0);

            % Store the CW background color as a special color pref
            % This way, if the CW bg color changes (via File/Preferences),
            % it will also affect existing rendered strs
            com.mathworks.services.Prefs.setColorPref('CW_BG_Color',xCmdWndView.getBackground);

            % Display the text in the Command Window
            count1 = fprintf(2,format,varargin{:});

            %awtinvoke(cmdWinDoc,'remove',lastPos,1);   % TODO: find out how to remove the extra '_'
            drawnow;   % this is necessary for the following to work properly (refer to Evgeny Pr in FEX comment
16/1/2011)
            docElement = cmdWinDoc.getParagraphElement(lastPos+1);
            if majorVersion<7 || (majorVersion==7 && minorVersion<13)
               if bolFlag && ~underlineFlag
                  % Set the leading hyperlink space character ('_') to the bg color, effectively hiding it
                  % Note: old Matlab versions have a bug in hyperlinks that need to be accounted for...
                  %disp(' '); dumpElement(docElement)
                  setElementStyle(docElement,'CW_BG_Color',1+underlineFlag,majorVersion,minorVersion);
%+getUrlsFix(docElement));
                  %disp(' '); dumpElement(docElement)
                  el(end+1) = handle(docElement);  %#ok used in debug only
               end

               % Fix a problem with some hidden hyperlinks becoming unhidden...
               fixHyperlink(docElement);
               %dumpElement(docElement);
```

```
    end

    % Get the Document Element(s) corresponding to the latest fprintf operation
    while docElement.getStartOffset < cmdWinDoc.getLength
       % Set the element style according to the current style
       %disp(' '); dumpElement(docElement)
       specialFlag = underlineFlag | boldFlag;
       setElementStyle(docElement,style,specialFlag,majorVersion,minorVersion);
       %disp(' '); dumpElement(docElement)
       docElement2 = cmdWinDoc.getParagraphElement(docElement.getEndOffset+1);
       if isequal(docElement,docElement2),  break;  end
       docElement = docElement2;
       %disp(' '); dumpElement(docElement)
    end

    % Force a Command-Window repaint
    % Note: this is important in case the rendered str was not '\n'-terminated
    xCmdWndView.repaint;

    % The following is for debug use only:
    el(end+1) = handle(docElement);  %#ok used in debug only
    %elementStart  = docElement.getStartOffset;
    %elementLength = docElement.getEndOffset - elementStart;
    %txt = cmdWinDoc.getText(elementStart,elementLength);
  end

  if nargout
     count = count1;
  end
  return;  % debug breakpoint

% Process the requested style information
function [underlineFlag,boldFlag,style] = processStyleInfo(style)
  underlineFlag = 0;
  boldFlag = 0;

  % First, strip out the underline/bold markers
  if ischar(style)
     % Styles containing '-' or '_' should be underlined (using a no-target hyperlink hack)
     %if style(1)=='-'
     underlineIdx = (style=='-') | (style=='_');
     if any(underlineIdx)
        underlineFlag = 1;
        %style = style(2:end);
        style = style(~underlineIdx);
     end

     % Check for bold style (only if not underlined)
     boldIdx = (style=='*');
     if any(boldIdx)
        boldFlag = 1;
        style = style(~boldIdx);
     end
     if underlineFlag && boldFlag
        warning('YMA:cprintf:BoldUnderline','Matlab does not support both bold & underline')
     end

     % Check if the remaining style sting is a numeric vector
     %styleNum = str2num(style); %#ok<ST2NM>  % not good because style='text' is evaled!
```

```
    %if ~isempty(styleNum)
    if any(style==' ' | style==',' | style==';')
      style = str2num(style); %#ok<ST2NM>
    end
  end

  % Style = valid matlab RGB vector
  if isnumeric(style) && length(style)==3 && all(style<=1) && all(abs(style)>=0)
    if any(style<0)
      underlineFlag = 1;
      style = abs(style);
    end
    style = getColorStyle(style);

  elseif ~ischar(style)
    error('YMA:cprintf:InvalidStyle','Invalid style - see help section for a list of valid style values')

  % Style name
  else
    % Try case-insensitive partial/full match with the accepted style names
    validStyles = {'Text','Keywords','Comments','Strings','UnterminatedStrings','SystemCommands','Errors', ...
                   'Black','Cyan','Magenta','Blue','Green','Red','Yellow','White', ...
                   'Hyperlinks'};
    matches = find(strncmpi(style,validStyles,length(style)));

    % No match - error
    if isempty(matches)
      error('YMA:cprintf:InvalidStyle','Invalid style - see help section for a list of valid style values')

    % Too many matches (ambiguous) - error
    elseif length(matches) > 1
      error('YMA:cprintf:AmbigStyle','Ambiguous style name - supply extra characters for uniqueness')

    % Regular text
    elseif matches == 1
      style = 'ColorsText';  % fixed by Danilo, 29/8/2011

    % Highlight preference style name
    elseif matches < 8
      style = ['Colors_M_' validStyles{matches}];

    % Color name
    elseif matches < length(validStyles)
      colors = [0,0,0; 0,1,1; 1,0,1; 0,0,1; 0,1,0; 1,0,0; 1,1,0; 1,1,1];
      requestedColor = colors(matches-7,:);
      style = getColorStyle(requestedColor);

    % Hyperlink
    else
      style = 'Colors_HTML_HTMLLinks';  % CWLink
      underlineFlag = 1;
    end
  end

% Convert a Matlab RGB vector into a known style name (e.g., '[255,37,0]')
function styleName = getColorStyle(rgb)
  intColor = int32(rgb*255);
  javaColor = java.awt.Color(intColor(1), intColor(2), intColor(3));
  styleName = sprintf('[%d,%d,%d]',intColor);
```

```
            com.mathworks.services.Prefs.setColorPref(styleName,javaColor);

        % Fix a bug in some Matlab versions, where the number of URL segments
        % is larger than the number of style segments in a doc element
        function delta = getUrlsFix(docElement)  %#ok currently unused
            tokens = docElement.getAttribute('SyntaxTokens');
            links  = docElement.getAttribute('LinkStartTokens');
            if length(links) > length(tokens(1))
                delta = length(links) > length(tokens(1));
            else
                delta = 0;
            end

        % fprintf(2,str) causes all previous '_'s in the line to become red - fix this
        function fixHyperlink(docElement)
            try
                tokens = docElement.getAttribute('SyntaxTokens');
                urls   = docElement.getAttribute('HtmlLink');
                urls   = urls(2);
                links  = docElement.getAttribute('LinkStartTokens');
                offsets = tokens(1);
                styles  = tokens(2);
                doc = docElement.getDocument;

                % Loop over all segments in this docElement
                for idx = 1 : length(offsets)-1
                    % If this is a hyperlink with no URL target and starts with ' ' and is collored as an error (red)...
                    if strcmp(styles(idx).char,'Colors_M_Errors')
                        character = char(doc.getText(offsets(idx)+docElement.getStartOffset,1));
                        if strcmp(character,' ')
                            if isempty(urls(idx)) && links(idx)==0
                                % Revert the style color to the CW background color (i.e., hide it!)
                                styles(idx) = java.lang.String('CW_BG_Color');
                            end
                        end
                    end
                end
            catch
                % never mind...
            end

        % Set an element to a particular style (color)
        function setElementStyle(docElement,style,specialFlag, majorVersion,minorVersion)
            %global tokens links urls urlTargets  % for debug only
            global oldStyles
            if nargin<3,  specialFlag=0;  end
            % Set the last Element token to the requested style:
            % Colors:
            tokens = docElement.getAttribute('SyntaxTokens');
            try
                styles = tokens(2);
                oldStyles{end+1} = styles.cell;

                % Correct edge case problem
                extraInd = double(majorVersion>7 || (majorVersion==7 && minorVersion>=13));  % =0 for R2011a-, =1 for
R2011b+
                %{
                if ~strcmp('CWLink',char(styles(end-hyperlinkFlag))) && ...
                    strcmp('CWLink',char(styles(end-hyperlinkFlag-1)))
```

```
    extraInd = 0;%1;
  end
  hyperlinkFlag = ~isempty(strmatch('CWLink',tokens(2)));
  hyperlinkFlag = 0 + any(cellfun(@(c)(~isempty(c)&&strcmp(c,'CWLink')),tokens(2).cell));
  %}

  styles(end-extraInd) = java.lang.String('');
  styles(end-extraInd-specialFlag) = java.lang.String(style);  %#ok apparently unused but in reality used by Java
  if extraInd
    styles(end-specialFlag) = java.lang.String(style);
  end

  oldStyles{end} = [oldStyles{end} styles.cell];
catch
  % never mind for now
end

% Underlines (hyperlinks):
%{
links = docElement.getAttribute('LinkStartTokens');
if isempty(links)
  %docElement.addAttribute('LinkStartTokens',repmat(int32(-1),length(tokens(2)),1));
else
  %TODO: remove hyperlink by setting the value to -1
end
%}

% Correct empty URLs to be un-hyperlinkable (only underlined)
urls = docElement.getAttribute('HtmlLink');
if ~isempty(urls)
  urlTargets = urls(2);
  for urlIdx = 1 : length(urlTargets)
    try
      if urlTargets(urlIdx).length < 1
        urlTargets(urlIdx) = [];  % '' => []
      end
    catch
      % never mind...
      a=1;  %#ok used for debug breakpoint...
    end
  end
end

% Bold: (currently unused because we cannot modify this immutable int32 numeric array)
%{
try
  %hasBold = docElement.isDefined('BoldStartTokens');
  bolds = docElement.getAttribute('BoldStartTokens');
  if ~isempty(bolds)
    %docElement.addAttribute('BoldStartTokens',repmat(int32(1),length(bolds),1));
  end
catch
  % never mind - ignore...
  a=1;  %#ok used for debug breakpoint...
end
%}

return;  % debug breakpoint
```

```
% Display information about element(s)
function dumpElement(docElements)
 %return;
 numElements = length(docElements);
 cmdWinDoc = docElements(1).getDocument;
 for elementIdx = 1 : numElements
   if numElements > 1,  fprintf('Element #%d:\n',elementIdx);  end
   docElement = docElements(elementIdx);
   if ~isjava(docElement),  docElement = docElement.java;  end
   %docElement.dump(java.lang.System.out,1)
   disp(' ');
   disp(docElement)
   tokens = docElement.getAttribute('SyntaxTokens');
   if isempty(tokens),  continue;  end
   links = docElement.getAttribute('LinkStartTokens');
   urls  = docElement.getAttribute('HtmlLink');
   try bolds = docElement.getAttribute('BoldStartTokens'); catch, bolds = []; end
   txt = {};
   tokenLengths = tokens(1);
   for tokenIdx = 1 : length(tokenLengths)-1
     tokenLength = diff(tokenLengths(tokenIdx+[0,1]));
     if (tokenLength < 0)
       tokenLength = docElement.getEndOffset - docElement.getStartOffset - tokenLengths(tokenIdx);
     end
     txt{tokenIdx}                                                                                                    =
cmdWinDoc.getText(docElement.getStartOffset+tokenLengths(tokenIdx),tokenLength).char;  %#ok
   end
   lastTokenStartOffset = docElement.getStartOffset + tokenLengths(end);
   txt{end+1}          =          cmdWinDoc.getText(lastTokenStartOffset,          docElement.getEndOffset-
lastTokenStartOffset).char;  %#ok
   %cmdWinDoc.uiinspect
   %docElement.uiinspect
   txt = strrep(txt',sprintf('\n'),'\n');
   try
     data = [tokens(2).cell m2c(tokens(1)) m2c(links) m2c(urls(1)) cell(urls(2)) m2c(bolds) txt];
     if elementIdx==1
       disp('    SyntaxTokens(2,1) - LinkStartTokens - HtmlLink(1,2) - BoldStartTokens - txt');
       disp('
================================================================================');
     end
   catch
     try
       data = [tokens(2).cell m2c(tokens(1)) m2c(links) txt];
     catch
       disp([tokens(2).cell m2c(tokens(1)) txt]);
       try
         data = [m2c(links) m2c(urls(1)) cell(urls(2))];
       catch
         % Mtlab 7.1 only has urls(1)...
         data = [m2c(links) urls.cell];
       end
     end
   end
   disp(data)
 end

% Utility function to convert matrix => cell
function cells = m2c(data)
 %datasize = size(data);  cells = mat2cell(data,ones(1,datasize(1)),ones(1,datasize(2)));
```

```
            cells = num2cell(data);

        % Display the help and demo
        function showDemo(majorVersion,minorVersion)
            fprintf('cprintf displays formatted text in the Command Window.\n\n');
            fprintf('Syntax: count = cprintf(style,format,...);  click <a href="matlab:help cprintf">here</a> for details.\n\n');
            url = 'http://UndocumentedMatlab.com/blog/cprintf/';
            fprintf(['Technical description: <a href="' url '">' url '</a>\n\n']);
            fprintf('Demo:\n\n');
            boldFlag = majorVersion>7 || (majorVersion==7 && minorVersion>=13);
            s = ['cprintf("text",    "regular black text");' 10 ...
                'cprintf("hyper",   "followed %s","by");' 10 ...
                'cprintf("key",     "%d colored",' num2str(4+boldFlag) ');' 10 ...
                'cprintf("-comment","& underlined");' 10 ...
                'cprintf("err",     "elements:\n");' 10 ...
                'cprintf("cyan",    "cyan");' 10 ...
                'cprintf("_green",  "underlined green");' 10 ...
                'cprintf(-[1,0,1],  "underlined magenta");' 10 ...
                'cprintf([1,0.5,0], "and multi-\nline orange\n");' 10];
            if boldFlag
                % In R2011b+ the internal bug that causes the need for an extra space
                % is apparently fixed, so we must insert the sparator spaces manually...
                % On the other hand, 2011b enables *bold* format
                s = [s 'cprintf("*blue",   "and *bold* (R2011b+ only)\n");' 10];
                s = strrep(s, '")',' ")');
                s = strrep(s, '",5)',' ",5)');
                s = strrep(s, '\n ','\n');
            end
            disp(s);
            eval(s);


        %%%%%%%%%%%%%%%%%%%%%%%% TODO %%%%%%%%%%%%%%%%%%%%%%%%
        % - Fix: Remove leading space char (hidden underline '_')
        % - Fix: Find workaround for multi-line quirks/limitations
        % - Fix: Non-\n-terminated segments are displayed as black
        % - Fix: Check whether the hyperlink fix for 7.1 is also needed on 7.2 etc.
        % - Enh: Add font support
```