LAUREA
AMMATTIKORKEAKOULU
Yhdessä enemmän

# Tilastografiikan tuotannon automaatio R-ohjelmointikielellä -

# Case Suomen Pankki

Aino Ojala

2018 Laurea

**Laurea-ammattikorkeakoulu**

# Tilastografiikan tuotannon automaatio R-ohjelmointikielellä - Case Suomen Pankki

Aino Ojala
Tietojenkäsittelyn koulutus
Opinnäytetyö
Joulukuu, 2018

**Laurea-ammattikorkeakoulu** **Tiivistelmä**
Tietojenkäsittelyn koulutusTietojenkäsittelyn koulutus
Tradenomi (AMK)

Aino Ojala

**Tilastografiikan tuotannon automaatio R-ohjelmointikielellä – Case Suomen Pankki**

Vuosi 2018 Sivumäärä 57

Tämän toiminnallisen opinnäytetyön tilaaja oli Suomen Pankki. Kehitysprojektin toimeksiantona oli tilastografiikkaa tuottava automaatioratkaisu R-ohjelmointikielellä. Opinnäytetyönä toteutettiin vakiomuotoista tilastografiikkaa tuottava ohjelma, joka noutaa tarvittavat tiedot, piirtää niistä tilastografiikkaa ja tallentaa kuvat. Raportissa kuvataan tämän ohjelman toimintaa ja kehitysvaiheita.

Ohjelman kehittämisessä pyrittiin automatisoimaan prosessi mahdollisimman pitkälle ja sisällyttämään siihen kaikki loppukäyttäjän toivomat ominaisuudet. Automatisoidusta tilastografiikasta pyrittiin saamaan visuaalisesti samanlaista kuin aikaisemmassa manuaalisessa ratkaisussa. Projektin aikana testattiin useita ominaisuuksia ja toteutusvaihtoehtoja, joista osa karsiutui lopullisesta ohjelmasta.

Lopputuotoksena toteutettiin ohjelma, joka hakee kuukausittain päivitettävät tiedot Suomen Pankin omasta tietokannasta, piirtää niistä tilastografiikkaa ja tallentaa kuvat kootuksi esitykseksi. Koska ratkaisua tullaan vielä tulevaisuudessa kehittämään, myös jatkokehityksen mahdollistaminen huomioitiin projektissa.

Opinnäytetyö kuvaa toteutettua ohjelmaa, sen toimintoja ja kehitysvaiheita. Lisäksi työssä esitellään tilastografiikan käsitteitä yleisellä tasolla ja käytäntöjä sekä R:n toimintoja. Raportissa käsitellään myös saatua palautetta ja arvioidaan lopputuotosta toimeksiannon alkuperäisten tavoitteiden pohjalta.

Asiasanat: Tilastografiikka, R-ohjelmointi, automaatio, Suomen Pankki

**Laurea University of Applied Sciences** **Abstract**
Degree Programme in Business Information Technology
Bachelor's Thesis

Aino Ojala

Automation of Statistical Graphics Process Using the Programming Language R — A Case Study of Bank of Finland

Year 2018 Pages 57

This thesis discusses an automation project which was conducted for the Bank of Finland in the summer 2018. The task was to develop an automated solution for standard statistical graphics production used in internal communication with the programming language R. In the development of the program the aim was to automate the process as much as possible, as well as to include all the features requested by the end-users. There was also an objective to make the automated statistical graphics visually similar to the graphics produced with the previous, manual solution. The final product is a program which fetches the monthly updated data from the Bank of Finland's own database, produces statistical graphics using the data, and saves the images as a complete presentation. Because the program will be developed further in the future, the means to make it possible were taken into account during the project.

The thesis will describe this program, the features it includes as well as its stages of development. The thesis will also present concepts and conventions related to statistical graphics, as well as R's functions at a general level, so that the descriptions in the report can be placed in the right context. The report will also discuss the feedback received and evaluate the final product using the original objectives of the project as the basis.

Keywords: Statistical graphics, R programming, automation, Bank of Finland

Sisällys

# 1 Johdanto

Tämän toiminnallisen opinnäytetyön tilaaja on Suomen Pankki. Kehitysprojektin toimeksiantona on tilastografiikkaa tuottava automaatioratkaisu R-ohjelmointikielellä. Opinnäytetyönä toteutetaan vakiomuotoista tilastografiikkaa tuottava ohjelma, joka noutaa tarvittavat tiedot, piirtää niistä tilastografiikkaa ja tallentaa kuvat. Raportissa kuvataan tämän ohjelman toimintaa ja kehitysvaiheita.

Ensin esitellään projektin toimeksianto ja tavoite. Tämän jälkeen siirrytään tietoperustaan, jossa käydään läpi tilastografiikan käsitteitä yleisellä tasolla. Lisäksi tarkastellaan sellaisia käytäntöjä, joilla saadaan luotua visuaalisesti ja informaatioarvoltaan onnistunutta tilastografiikkaa. Tietoperustassa on myös kuvattu R-ohjelmointikieltä ja sen toimintaa avaamalla sen tarkoitusta ja toiminnallisuutta. Lisäksi käsitellään R:n monipuolisia datankäsittely- ja visualisointiominaisuuksia, ja miten sitä voi laajentaa käyttäjäyhteisön koodaamilla paketeilla.

Ohjelmaa käsittelevässä osassa kuvataan toteutetun ohjelman toimintaa kokonaisuutena ja esitellään sen tarkoitus. Lisäksi perustellaan sen tuottamaan tilastografiikkaan liittyviä ratkaisuja ja käydään läpi käytännöt, joiden mukaan piirrettävä grafiikka on tehty. Tämän jälkeen ohjelman ominaisuuksia käsitellään yksityiskohtaisemmin. Kuvauksessa on mainittu ohjelman kannalta tärkeimpiä komentoja, sekä selitetty miten ne vaikuttavat ohjelman toimintaan. Lopuksi esitellään erikseen ohjelman ominaisuuksia, joiden kehittämisprosessi oli hieman muita pidempi. Osaan ominaisuuksien vaiheista sisältyi vaihtoehtoisia toteutustapoja ja kompromisseja. Näitä on käyty raportissa läpi.

Automaatioratkaisua jatkokehitetään tulevaisuudessa. Jatkokehitys on otettu huomioon projektin aikana kuvaamalla kehitystyön mahdollistamiseen ja helpottamiseen liittyviä ratkaisuja.

# 2 Tilastografiikan tuotannon automaatio

Opinnäytetyön aiheena on Suomen Pankin tilastoyksikölle toteutettu tilastografiikan automaatioratkaisu. Opinnäytetyöraportissa esitellään lopputuotoksena tehty ohjelma, sen toiminta, ominaisuudet ja kehitysvaiheita. Opinnäytetyössä pyritään myös antamaan lukijalle pohjatietoa tilastografiikasta ja ohjelman toteuttamiseen käytetystä R-ohjelmointikielestä.

## 2.1 Taustat

Suomen Pankin sisäisessä viestinnässä käytettävän niin kutsutun ”Slaidisetin” kuukausittainen päivitysprosessi halutaan automatisoida. Slaidisetti on koottu esitys, joka sisältää monipuolisesti tilastografiikkaa Suomen ja Euroalueen maiden taloustilanteiden mittareista. Slaidisetti luodaan PowerPointissa, ja tilastografiikka päivitetään manuaalisesti hakemalla kuukausittain

päivittyvää dataa Suomen Pankin omista aikasarjoista. Noin 50-diaisen Slaidisetin päivittämiseen kuluu tällä hetkellä suunnilleen kaksi työpäivää kuussa. Automaatioratkaisulla sama työ saataisiin tehtyä minuuteissa, joten ratkaisu siis säästäisi paljon aikaa ja vaivaa.

## 2.2 Tavoitteet

Käytännön projektin tavoitteena on ollut automatisoida Slaidisetin päivitysprosessi niin pitkälle kuin mahdollista. Tavoitteena oli kehittää R-koodilla toteutettu ohjelma, joka osaisi hakea tiedot, piirtää niistä halutun mallista ja julkaisuvalmista tilastografiikkaa, ja koota grafiikka valmiiksi esitykseksi. Tämä helpottaisi suuresti Slaidisetistä vastaavan työtaakkaa ja säästäisi työaikaa.

Olen myös itse asettanut tavoitteeksi tehdä automaatioratkaisusta sellaisen, että sitä on helppo jatkokehittää. Slaidisetin kuvaajia uudistetaan ja uusia kuvaajia lisätään esitykseen, joten on tärkeää, että koodi on helppolukuista, selkeää, ja että sitä on vaivatonta muokata. Tämän tavoitteen toteuttamiseksi olen laatinut myös ohjeet ohjelman käyttämiseen ja muokkaamiseen.

Itse opinnäytetyöraportin tavoite on kuvata tämän käytännön projektin tuotoksena syntynyttä ratkaisua, hieman siihen liittyvää työskentelyä ja erityisesti automatisoidun ratkaisun toimintaa. Erityisesti tarkoituksena on tuoda esille omia ratkaisujani, sekä perustelut ja kehitysvaiheet niiden takana.

## 2.3 Työskentely

Työskentelin käytännön projektin parissa Suomen Pankin tiloissa täyspäiväisesti toukokuun lopusta elokuun alkuun. Käytössäni oli tarvittava ohjelmisto ja pääsy Suomen Pankin sisäiseen verkkoon. Harjoiteltuani R:n perusteet siirryin suoraan projektin työstämiseen.

Projekti on ollut hyvin käytännönläheinen, ja oma osuuteni on ollut pitkälti koodin tuottamista. Projektille on asetettu toimeksiantajan taholta selkeät raamit jo alusta lähtien, kuten esimerkiksi R-ohjelmointikieli ratkaisun toteutusalustana, ja valmiin tilastografiikan visuaalinen ilme pääpiirteissään. Myös ohjelman rakennetta oltiin jo suunniteltu ennen liittymistäni projektiin, ja koodi sisältääkin osia, joita en ole itse tuottanut. Koodaamisen lomassa olen kuitenkin saanut melko vapaat kädet toivottujen ominaisuuksien toteuttamisen ideoinnissa, sekä koodiin liittyvien solmukohtien ratkomisessa.

Työskentely ei ole edennyt minkään varsinaisen suunnitelman mukaan. Harjoiteltuani R-koodin kirjoittamista ja ajamista RStudiolla hieman siirryin kehittämään ja testaamaan ohjelmaa pala kerrallaan. Projektin varrella käydyissä keskusteluissa verkkojulkaisutiimin ja tilastoyksikön työntekijöiden kanssa syntyi myös uusia ideoita ja kehitysehdotuksia, joiden pohjalta pyrin sisällyttämään ohjelmaan uusia ominaisuuksia.

Omasta työskentelystäni riippumattomista syistä ohjelma ei ole vielä täysin käyttövalmis. Tämä kyettiin ennakoimaan jo projektin aikana, joten panostin ohjelman kehittämisen ohella myös jatkokehityksen mahdollistamiseen ja helpottamiseen. Opinnäytetyössä kuvaillaan jatkokehitykseen liittyviä syitä, ratkaisuja ja työvaiheita.

# 3 Tilastografiikka

Vaikka työskentely opinnäytetyöprojektissa on keskittynytkin suurimmaksi osaksi koodin tuottamiseen, on tilastografiikka ollut tärkeässä roolissa. Tässä luvussa käsitellään yleisesti tilastografiikkaa, sen tarkoitusta, mistä se muodostuu, sekä mitkä ovat tilastografiikan hyviä käytäntöjä. Automaatioratkaisun tuottama tilastografiikka muodostuu tässä luvussa esitellyistä osista ja nojaa pitkälti käsiteltäviin käytäntöihin.

## 3.1 Tilastografiikan määritelmä

Tilastografiikka on nimensä mukaisesti grafiikkaa, jolla esitetään tilastotietoa graafisessa muodossa. Tilastografiikan tarkoitus on tuoda tilasto helpommin havainnoitavaan ja muotoon, jolloin sen sisältämä tieto myös välittyy tehokkaammin (Karjalainen & Karjalainen 2009, 14). Vaikka tilastografiikka ei välttämättä sovikaan tarkkojen lukujen esittämiseen, se mahdollistaa tilaston arvojen helpomman vertailun. Tilastografiikka on myös huomattavasti mielenkiintoisempi ja huomiota herättävämpi tapa esittää tietoa. Onnistuneen tilastografiikan tulisi antaa sitä tarkastelevalle oikea käsitys tiedosta yhdellä silmäyksellä (Holopainen & Pulkkinen 2012, 53). Jotta tilastografiikan hyödyt kuitenkin saataisiin valjastettua, on sen toteutustapoja harkittava hyvin.

## 3.2 Tilastografiikan osat

Tilastograafinen kuvio eli kuvaaja muodostuu monesta osasta, jotka kukin edesauttavat tiedon välitystä. Vaikka erilaiset kuvaajatyypit saattavat erota ulkonäöltään toisistaan, ovat niiden rakenteet pitkälti samanlaiset. Tässä luvussa käydään läpi tilastografiikan yleistä rakennetta.

Nimensä mukaisesti tilastografiikka sisältää grafiikkaa, joka kuvaa tilastotietoa. Tietoa voidaan kuvata visuaalisesti lukuisilla erilaisilla tavoilla, mutta tärkeintä on, että grafiikka havainnollistaa sitä totuudenmukaisesti. Grafiikan elementtien tulee siis kuvastaa ominaisuuksillaan, kuten pituudella ja pinta-alalla, kuvattavien datan osien todellista suhdetta (Karjalainen & Karjalainen 2009, 16).


Kuvio 1 Viivakuvaaja yleisön lainakannan vuosikasvusta Suomessa ja Euroalueella

Opinnäytetyöprojektissa on työskennelty sellaisen tilastografiikan parissa, jossa datan sisältämät havainnot asetetaan x- ja y-akseleille. Kaikki kuvaajatyypit, esimerkiksi piirakkakuvaaja, eivät käytä akseleita datan kuvaamiseen. Yleensä kuitenkin esimerkiksi tietyllä aikavälillä tapahtuvia muutoksia, eli aikasarjaa, kuvaava tilastografiikka sisältää akselit. Kummatkin akselit ovat asteikkoja, jotka mittaavat jotakin käytettävän datan muuttujaa. Esimerkkikuvassa y-akseli mittaa havainnon arvoa, eli tässä tapauksessa vuosikasvua. X-akselilla taas mitataan havainnon ajankohtaa. Näiden asteikkojen muodostamaan akseliruudukkoon piirretään datan mukainen kuvaaja arvon ja päivämäärän suhteesta: syntyy tilastografiikkaa, joka kertoo, mikä havainnon arvo on ollut minäkin ajankohtana. Kun havaintoja on paljon ja pitkältä aikaväliltä, kuten esimerkkikuvaajassa, voidaan visuaalisesti hahmottaa havaintojen kehitystä. Akseleita voi käyttää mittaamaan mitä tahansa datan muuttujia, joten ne soveltuvat myös esimerkiksi korrelaation kuvaamiseen.

Pelkästään graafinen esitys ei luonnollisesti riitä, vaan se tarvitsee myös tekstiä selkeyttämään kuvattavaa dataa (Karjalainen & Karjalainen 2009, 16). Tilastografiikka sisältääkin yleensä ainakin otsikon ja selitteet. Akseleita käyttävissä kuvaajissa akseleilla tulee luonnollisesti olla asteikko ja otsikko sille, mitä ne mittaavat. Kuvaajat voivat sisältää myös alaotsikon ja kuvatekstin, joista jälkimmäisessä yleensä mainitaan tilaston lähde. Tekstin ensisijainen tarkoitus on kertoa kuvaajan tarkastelijalle, mistä siinä on kyse. Kuten graafisen osan, myös tekstin on pyrittävä antamaan tarkastelijalle oikea kuva siinä kuvatuista tiedoista.

Esimerkkikuvaan on piirretty viivakuvaaja kahden alueen lainakannan vuosikasvusta. Vaikka otsikko kertookin, mistä alueista on kysymys, vasta sen alapuolelle sijoitettu selite kertoo, kumpi viiva kuvastaa kutakin aluetta. Selite on siis välttämätön osa sellaista tilastografiikkaa, johon sisältyy esimerkkikuvaajan tavoin eroteltuja havaintoja. Jos esimerkkikuvaaja sen sijaan kuvaisi vain Suomea, selite ei olisi tarpeellinen. Selitteeseen kuuluu väri, joka edustaa mitattavaa asiaa, sekä teksti, jossa mitattava asia kuvataan lyhyesti. Pistekuvaajat saattavat käyttää värien sijaan erilaisia kuvioita kuvaamaan eriteltyjä tietoja (Karjalainen & Karjalainen 2009, 29).

## 3.3 Erilaiset kuvaajatyypit ja niiden käyttötarkoitukset

Tilastografiikan käsitteeseen mahtuu suuri määrä toisistaan eroavia graafisia tapoja esittää tietoa. Vaikka tilastografiikan luominen tiedonkäsittelyyn erikoistuvilla ohjelmilla valmiin datan pohjalta voi olla nopeaa ja vaivatonta, kuvaajien tekemisessä on kuitenkin hyvä käyttää harkintaa (Karjalainen & Karjalainen 2009, 14). Erityisen tärkeää on pohtia, millainen kuvaajatyyppi ylipäänsä sopii datalle, joka halutaan esittää. Valittu kuvaajatyyppi vaikuttaa paljon siihen, miten tieto kuvaajan kautta välittyy. Jos kuvaajatyyppi ei sovi käytettävän datan kuvaamiseen, voi tiedon havainnointi olla jopa haastavaa. Tässä luvussa käydään läpi erilaisia yleisimpiä tilastografiikan tyyppejä, ja kerrotaan, millaisen datan kuvaamiseen ne soveltuvat parhaiten.

### 3.3.1 Pylväskuvaajat

Pylväskuvaajissa nimensä mukaisesti tietoa kuvataan pylväillä. Pylväät voidaan asetella joko pysty- tai vaakatasoon riippuen siitä, millaista tietoa niillä halutaan kuvata. Pylväskuvaajia käytettäessä on tärkeää, että pylväät esitetään kokonaisena, eli akselin asteikko alkaa nollasta. Jos asteikko katkaistaan, eivät piirrettävät pylväät enää kuvasta tietoja oikeassa suhteessa (Karjalainen & Karjalainen 2009, 19).

Slaidisetissä on käytetty pystysuuntaisia pylväitä kuvaamaan erilaisia kantoja tietyllä aikavälillä. Pystysuuntaiset pylväät sopivatkin parhaiten kuvaamaan määriä sekä niiden muutoksia. (Karjalainen & Karjalainen 2009, 19). Kuvaajan y-akselille annetaan määräasteikko, jolla mitataan muuttujaa. X-akselia taas käytetään yleensä ajan mittaamiseen, mutta sitä voi käyttää myös luokittelevana asteikkona, jos luokkia on vain muutama (Hildén, Koponen & Vapaasalo 2017, 186).

Luokkien perusteella eriteltyjen tietojen mittaamiseen sopii kuitenkin muissa tapauksissa paremmin vaakasuuntainen pylväskuvaaja, jossa määräasteikko onkin x-akselilla, ja y-akseli sisältää päällekkäisen listan eri luokista. Luokkien otsikot mahtuvat kuvaajaan paremmin, kun ne on esitelty allekkain. Pystysuuntaisissa kuvissa palkkien välille mielletään myös helposti jatkumo, mikä voi olla harhaanjohtavaa. Allekkaisesta listasta ei synny tällaista väärää kuvaa

luokitellusta datasta (Hildén, Koponen & Vapaasalo 2017, 186). Luokkien järjestys voi vaihdella, mutta yleensä pylväät luokkineen asetellaan suuruusjärjestykseen pisimmästä alkaen (Karjalainen & Karjalainen 2009, 23).

Pylväskuvaajissa pylväitä voidaan asetella myös päällekkäin, kuten joissain Slaidisetin kuvissa on tehtykin. Päällekkäisten pylväiden kuvaajien on tarkoitus kuvata kokonaisuus, sekä osatekijöiden osuus siitä (Hildén, Koponen & Vapaasalo 2017, 187). Päällekkäisiä pylväitä voidaan käyttää niin pysty- kuin vaakasuuntaisissa pylväskuvaajissa.

### 3.3.2 Pistekuvaajat

Pistekuvaajissa kuvataan kahden muuttujan välistä suhdetta. Akseleiden määräasteikolla mitataan kahta eri muuttujaa, ja havainto asetetaan ruudukolle molempia akseleita käyttäen. (Hildén, Koponen & Vapaasalo 2017, 194). Tämä tietenkin edellyttää, että havainnolla on jokin arvo molemmille muuttujille: että esimerkiksi pituuden ja painon suhdetta vertaavaa kuvaajaa varten tiedetään sille asetettavan havaintohenkilön pituus ja paino. Toisin kuin palkkikuvaajien tapauksessa, pistekuvaajien akseleita voi katkoa, mutta tämä on hyvä merkitä selkeästi (Karjalainen & Karjalainen 2009, 29). Slaidisetti ei sisällä tällaisia kuvaajia, joten niistä on kerrottu tässä vain suppeasti.

### 3.3.3 Viivakuvaaja

Viivakuvaajat toimivat samaan tapaan kuin pistekuvaajat, mutta x-akselin asteikolla on oltava tasavälinen muuttuja, kuten esimerkiksi vuosiluku tai kuukausi (Karjalainen & Karjalainen 2009, 32). Viivakuvaaja soveltuu hyvin kehityksen ja jatkuvien ilmiöiden mittaamiseen ja niitä käytetäänkin Slaidisetissä esimerkiksi korkojen ja vuosikasvun kuvaamiseen.

Viivakuvaaja perustuu yleensä pistekuvaajaan, jossa havaintopisteet on yhdistetty viivalla. (Hildén, Koponen & Vapaasalo 2017, 190) Yhtenäisestä viivasta huolimatta viivakuvaaja ei siis kerro mitattavan muuttujan arvoa jokaisena hetkenä, vaan tiettyinä havaintoaikoina. Esimerkiksi tilastografiikan osia käsittelevän luvun esimerkkikuvaaja muodostuu kerran kuukaudessa päivitettävistä tiedoista. Viivakuvaajaa kuitenkin voidaan pyöristää tai porrastaa riippuen käyttötarkoituksesta. Porrastuksella voidaan tuoda esiin äkillinen muutos, kun taas pyöristystä voidaan käyttää silloin, kun arvot perustuvat matemaattisiin tai luonnon lakeihin (Hildén, Koponen & Vapaasalo 2017, 190).

### 3.3.4 Piirakkakuvaaja

Piirakkakuvaaja tai ympyräkuvaaja kuvaa erilaisten osien osuutta yhdestä kokonaisuudesta. Nämä osiot eli sektorit ovat suhteellisia, ja usein ne perustuvatkin prosenttisuuksiin (Karjalainen & Karjalainen 2009, 27). Piirakkakuvaajaa pidetään yleisesi melko epätarkkana tapana esittää tietoa, joten se soveltuu lähinnä likimääräisten erojen kuvaamiseen (Hildén, Koponen & Vapaasalo 2017, 199). Sektoreita olisi hyvä olla selkeyden vuoksi maksimissaan kuusi, ja ne

tulisi esittää suuruusjärjestyksessä (Karjalainen & Karjalainen 2009, 27). Automaatioratkaisuun ei ole lisätty vielä piirakkakuvaajia, mutta sellaisia on manuaalisessa Slaidisetissä.

### 3.4 Tilastografiikan käytäntöjä

Aiemmissa luvuissa on jo hieman kerrottu siitä, mikä on tilastografiikan tarkoitus, sekä millaista on hyvä ja tarkoituksenmukainen tilastografiikka. Tässä luvussa esitellään tilastografiikassa käytettyjä yleisiä hyviä käytäntöjä, joita on hyödynnetty myös automaatioratkaisun tuottamissa kuvaajissa. Luvussa esitellään myös miten näiden käytäntöjen laiminlyöminen vaikuttaa tilastografiikan luettavuuteen.

Kuten aikaisemmin on kerrottu, tilastografiikan tulisi antaa lukijalle oikea käsitys tilaston tiedoista yhdellä silmäyksellä. Tämä periaate ulottuu kaikkiin tilastografiikan osiin aina kuvaajatyypin valinnasta tekstin sisältöön. Tilastografiikalle on siis tärkeää olla kaikilta osiltaan selkeää, totuudenmukaista ja helposti ymmärrettävää.

Oikeanlaisen kuvaajatyypin valinnasta on kerrottu kuvaajatyyppejä käsittelevässä luvussa. Jos valittu kuvaajatyyppi ei sovi käytettävälle aineistolle, voi kuvaaja olla hankalasti havainnoitavissa tai jopa harhaanjohtava. Esimerkiksi ympyräkaavio ei sovi eri alojen keskipalkkojen vertailuun, sillä nämä eivät muodosta kokonaisuutta (Karjalainen & Karjalainen 2009, 47). Pelkkä oikeanlaisen kuvaajatyypin valinta ei kuitenkaan takaa onnistunutta kuvaajaa, vaan selkeä toteutus vaatii useiden seikkojen huomioon ottamista.

Kuten kuvaajatyyppejä käsittelevässä luvussa tulee ilmi, erilaisia kuvaajatyyppejä tulee käsitellä eri tavoin. Esimerkiksi viiva- ja pistekuvioiden tapauksessa y-akselin voi katkaista niin, ettei se varsinaisesti alakaan nollasta, jos tämä vain sopii kuvattavaan aineistoon, ja on merkitty selkeästi. Näin ei kuitenkaan ole hyväksyttyä tehdä pylväskuvaajien tapauksessa, sillä ne perustuvat pituuksien vertailuun (Hildén, Koponen & Vapaasalo 2017, 220). Katkaistut pylväskuvaajat antavat harhaanjohtavan kuvan vertailtavien tietojen suhteesta, joten katkaisua voi käyttää harhaanjohtavasti saamaan tietojen erot näyttämään ensisilmäyksellä suuremmilta, kuin ne oikeasti ovat.

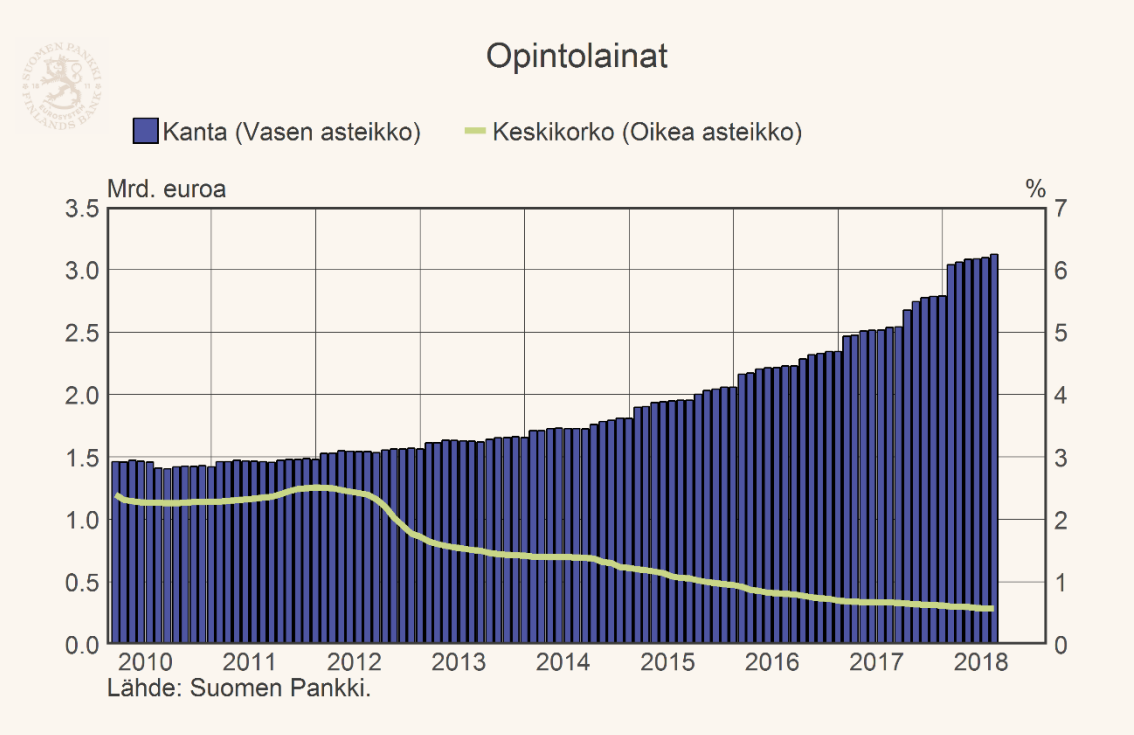
Itse kuvaajan on oltava myös graafisesti onnistunut. Selkeyden vuoksi on hyvä pysyä yksinkertaisessa toteutuksessa. Esimerkiksi kuvaajan osien erottelu toisistaan levottoman näköisellä kuvioinnilla voi tehdä kuvaajasta rasittavan katsoa (Karjalainen & Karjalainen 2009, 44). Myös kolmiulotteisuustehosteita on hyvä välttää niin pylväs- kuin ympyräkuvioissa: ne vääristävät mittasuhteita ja tekevät asteikkojen tulkinnasta hankalaa (Karjalainen & Karjalainen 2009, 38).

Kuvaajassa tekstit ovat yhtä tärkeitä kuin kuvaajan grafiikkakin, joten niiden sisältöä ja visuaalista toteutusta on myös harkittava hyvin. Sisällöltään otsikoiden, selitteiden, akseleiden nimien ja muihin kuvaajaan sisältyvien tekstien on kerrottava selkeästi, mistä kuvaajassa on

kysymys. Tekstien sisältö on hyvä muotoilla lyhyiksi ja ytimekkäiksi, sillä kaikilla kuvaajan osilla tulee olla informaatioarvoa (Karjalainen & Karjalainen 2009, 16). Asteikoille ei tule esimerkiksi kirjoittaa turhan paljoa tekstiä, sillä lopputulos voi olla vaikealukuinen (Karjalainen & Karjalainen 2009, 45).

Myös tekstien tapauksessa visuaalisella toteutuksella on painoarvoa. Tekstit tulisi aina mahdollisuuksien mukaan asetella vaakasuoraan (Karjalainen & Karjalainen 2009, 45). Myös koko kuvaajan visuaalinen ilme on väreiltään suunniteltava sellaiseksi, että tekstit on helppo lukea. Elementeillä tulisi olla toisiinsa nähden hyvä kontrasti, jotta tekstit erottuisivat selkeästi (Karjalainen & Karjalainen 2009, 42). Tilastografiikassa pätevät siis pitkälti samanlaiset säännöt, kuin graafisessa suunnittelussa yleensäkin.

Akselit ja niiden muodostama ruudukko vaikuttavat olennaisesti siihen, millaiselta niille asetettu tilastografiikka näyttää, ja miten tieto niissä korostuu (Karjalainen & Karjalainen 2009, 30). Asteikkojen tulisi olla tasavälisiä, jotta kuvaajan mittasuhteet eivät vääristy (Karjalainen & Karjalainen 2009, 56). Ruudukon visuaalisen toteutuksen tulisi olla sellainen, ettei se häiritse datan luettavuutta kuvasta (Karjalainen & Karjalainen 2009, 45). Ruudukon viivat voi esimerkiksi toteuttaa katkoviivoilla, haaleammalla värillä, tai ohuemmalla paksuudella.




Kuvio 2 Viiva- ja pylväskuvaaja opintolainojen kannasta ja keskikorosta

Joskus kuvaajissa voi olla tarpeen käyttää kahta y-akselia, joilla on omat asteikkonsa. Slaidisetti sisältää tällaisia kuvaajia, joten niiden toteutukseen liittyviä käytäntöjä käsitellään tässä

erikseen. Kahden y-akselin kuvaajassa on ensinnäkin tärkeää, että vastakkaiset asteikot käyttävät samaa ruudukkoa. Esimerkkikuvaajassa on käytetty kahta y-akselia, jotka käyttävät arvoiltaan erilaisia, mutta viivamäärältään samanlaisia asteikoita. Kahden y-akselin kuvaajassa on tuotava myös selkeästi esiin kumpaa akselia kukin piirretty data käyttää. Jos tämä on tuotu esille puutteellisesti tai epäselvästi, voi lopputulos olla hyvin harhaanjohtava. Esimerkkikuvaajassa tieto on sisällytetty selitteisiin, mutta myös värikoodaus on mahdollinen vaihtoehto asteikoiden käytön selkeyttämiseksi (Hildén, Koponen & Vapaasalo 2017, 219).

# 4 R

Projektissa tilastografiikan piirtämiseen on käytetty R-ohjelmointikielellä toteutettua ohjelmaa. Opinnäytetyöraportin kuvitus koostuu osittain juurikin tällä kyseisellä ohjelmalla piirretystä tilastografiikasta. Tässä luvussa esitellään R, sen toimintaa ja etuja tiedonkäsittelyn välineenä. Tarkoituksena on myös pohjustaa toiminnallista osuutta, joka sisältää paljon kuvausta toteutetun ohjelman toiminnasta. Koska R on todella laaja kokonaisuus, keskitytään tässä luvussa pitkälti vain niihin ominaisuuksiin ja toimintoihin, jotka ovat toiminnallisen projektin kannalta relevantteja.

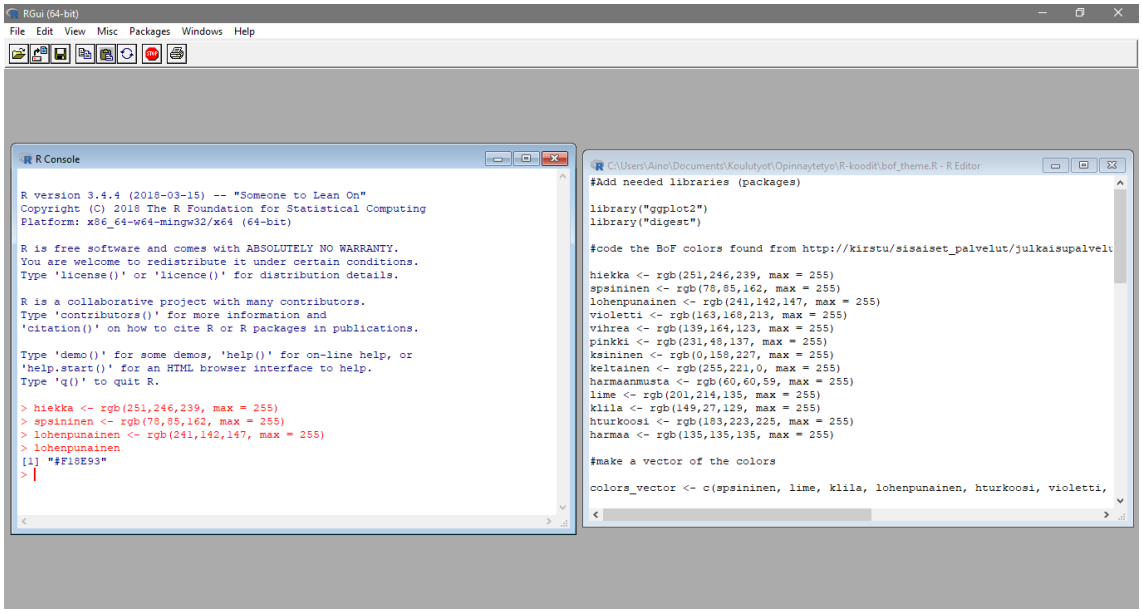
## 4.1 R ohjelmointikielenä

R on tilastolliseen tiedonkäsittelyyn ja grafiikkaan erikoistunut ohjelmointikieli ja toteutusympäristö (Gentleman & Ihaka 1996, 299; R-Project 2018a). R kykenee monipuoliseen datankäsittelyyn ja kuvaajien piirtämiseen. Se on täysin ilmainen ja lähdekoodiltaan vapaa, ja sitä on helppo laajentaa (R-Project 2018a). Laajennus tapahtuu paketeilla, jotka ovat usein R:n käyttäjäyhteisön tuottamia (Gohil 2015, 8). Paketit ja niiden sisältämät erilaiset funktiot tekevät R:stä hyvin monipuolisen välineen tilastografiikan tuottamiseen.

Ohjelmointikielen ovat alun perin kehittäneet Ross Ihaka ja Robert Gentleman, ja sen lähdekoodi julkaistiin vuonna 1995. R:ssä haluttiin yhdistää kahden olemassa olevan ohjelmointikielen hyödyllisimmät ominaisuudet ja luoda kieli tilastoanalyysille. Nämä ohjelmointikielet olivat S ja Scheme. R muistuttaa syntaksiltaan paljon S:ää, mutta toiminnot ovat lähempänä Schemeä (Gentleman & Ihaka 1996, 288). Tällä hetkellä R-projektin parissa työskentelee Ihankan ja Gentlemanin lisäksi useita myötävaikuttajia (R-Project 2018b).

## 4.2 R:n perustoiminnot

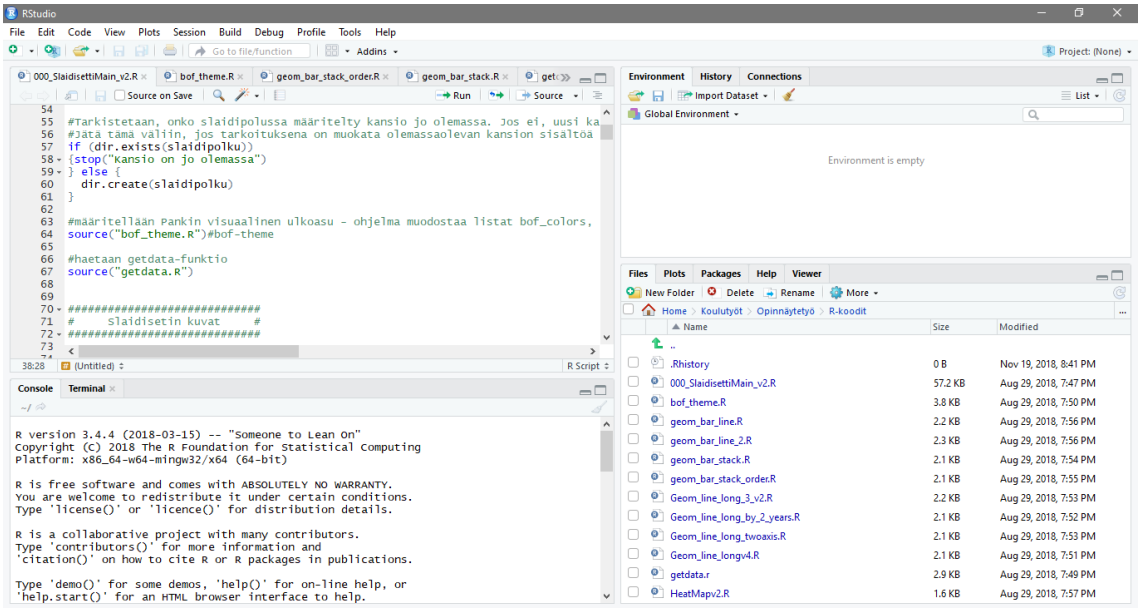
### 4.2.1 Käyttö




Kuvio 3 RGui-käyttöliittymä

R:n käyttö perustuu komentoihin (Gentleman & Ihaka 1996, 300). R:n RGui-käyttöliittymässä sen konsolille voi syöttää komentoja suoraan, tai ajaa valmiiksi kirjoitettua koodia R-tiedostoista. Projektissa R:ää on käytetty RStudio-käyttöliittymällä, joka sisältää konsoli- ja tiedostonäkymien lisäksi myös useita hyödyllisiä näkymiä, kuten tiedostot, piirrettyjen kuvaajien esikatselun ja listan käytössä olevista paketeista. R:n tapaan myös RStudio on ladattavissa ilmaiseksi. Käyttöliittymästä riippumatta R:n käyttö vaatii jonkin verran ohjelmointiosaamista,

etenkin jos tarkoituksena on koodin ajamisen lisäksi myös koodata uusia kuvaajia R:n piirrettäväksi.




Kuvio 4 RStudio-käyttöliittymä

Yksinkertaistettuna kuvaajan piirtämiseen R:llä vaaditaan siihen tarvittavan datan syöttämistä R:ään, sekä komentoa, joka piirtää kuvaajan. Mikäli kuvaaja halutaan tallentaa kuvatiedostona, tarvitaan siihen erillinen komento. Datankäsittelystä ja kuvaajan piirtämisestä kerrotaan myöhemmissä luvuissa tarkemmin.

Datankäsittelyn ja kuvaajien piirtämisen lisäksi R:llä on mahdollista määrittää omia funktioita, joilla voi nopeuttaa ohjelmointia ja yksinkertaistaa koodia. R sisältää luonnollisesti valmiita komentoja, mutta usein on myös tarpeen käyttää funktioita. Funktiota luodessa määritellään ikään kuin uuden komennon toiminnot, ja nimetään tämä kokonaisuus (Grolemund & Wickham 2017). Kun määrittely on tehty, funktiota voi käyttää koodissa kuten mitä tahansa komentoa. R:n ominaisuuksia laajentavat paketit perustuvat juurikin R:n käyttäjäyhteisön koodaamiin funktiokirjastoihin, jotka ovat vapaasti käytettävissä.

R-koodia voi yksinkertaistaa myös luomalla muuttujia. Muuttujiin sisällytetään jokin arvo, kuten vaikka lukujono, osoite, otsikko, tai mitä tahansa osa koodia. Muuttujia määritellään samalla tavalla kuin funktioita, ja kun määrittely on tehty, niitäkin voi käyttää koodissa viittaamalla niiden nimeen. Muuttujia voidaan käyttää esimerkiksi määrittelemään kuvaajan piirtämisessä käytettävien visuaalisten parametrien arvojen määrittelyyn, kuten automaatioprojektissa onkin tehty.

### 4.2.2 Datankäsittely

Tilastollisen tiedonkäsittelyn välineenä R luonnollisesti tarvitsee dataa käsitteleviä ominaisuuksia. R mahdollistaa datan tallentamisen ja käyttämisen erilaisilla datatyypeillä, sekä myös datan lukemisen ulkoisista lähteistä, kuten esimerkiksi Excel-tiedostoista tai SQL-haun kautta. Tässä luvussa käydään läpi R:n erilaisia datatyyppejä ja niiden käyttötarkoituksia.

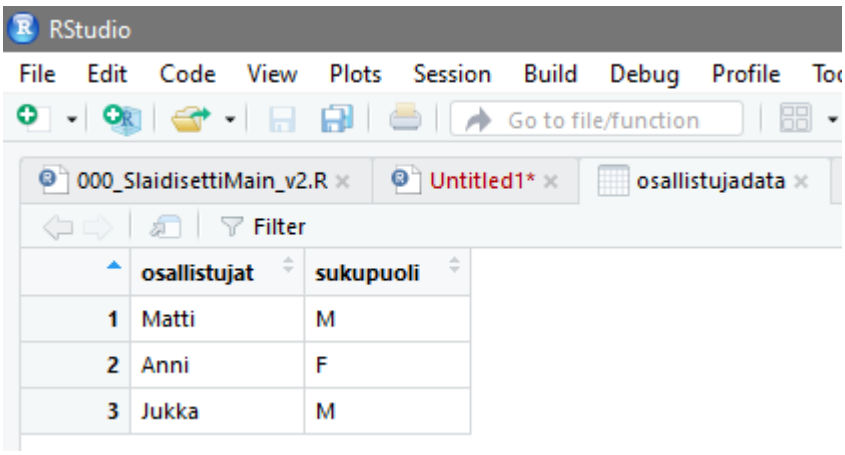
Yksi R:n perusdatatyypeistä ovat vektorit. Vektoreihin voi tallentaa dataa arvoina, joita kutsutaan elementeiksi. Samaan vektoriin voi tallentaa kuitenkin vain yhdentyyppistä dataa: esimerkiksi teksti- ja numerodataa ei voi tallentaa samaan vektoriin (Lantz 2013, 53). Vektori luodaan komennolla ”<-c()”. Esimerkiksi komennolla ”osallistujat <- c(”Matti”,”Anni”,”Jukka”)” luodaan ”osallistujat”-niminen vektori, joka sisältää ”c()”- komennon sisältämät nimet. Kun vektori on luotu, muut komennot voivat käyttää sen dataa. Esimerkkivektorin tietoja voitaisiin tarkastella esimerkiksi ”osallistujat[2]”, komennolla, jolloin R:n konsoli tulostaisi vektorin toisen arvon, eli nimen ”Anni”. Opinnäytetyössä vektoreita ei ole käytetty itse piirrettävän datan käsittelyyn, vaan kuvaajien visuaalisten ominaisuuksien säätämiseen.

Toinen R:n käyttämä datatyyppi on faktori. Faktoriin voidaan sisällyttää toistuvia arvoja, ja on tähän tarkoitukseen tehokkaampi datatyyppi kuin vektori (Lantz 2013, 55). Faktoriin voidaan esimerkiksi sisällyttää edellisen luvun esimerkin mukaisesti osallistujien sukupuolet: ”sukupuoli <- factor(c(”M”, ”F”, ”M”))”. Faktori ”sukupuoli” sisältää kolme havaintoa, mutta vain kahdentyyppisiä havaintoja, ”M” ja ”F”. Kun faktorin tulostaa, konsoli näyttää havaintojen lisäksi myös faktorin kaikki mahdolliset tasot. Esimerkiksi jos faktoriin olisi määritelty erillisellä levels-komennolla kolmas vaihtoehto ”X”, se ilmoitettaisiin mahdollisena tasona faktoria tulostettaessa, vaikkei faktori itse sisältäisi tätä arvoa. Vektorin tapaan myöskään faktoria ei ole käytetty opinnäytetyössä varsinaiseen piirrettävän tiedon käsittelyyn.

Lista on erityistyyppinen vektori, johon pystyy sisällyttämään useantyyppistä dataa. Tavallisesta vektorista poiketen listan elementtejä voi myös nimetä, mikä tekee datan käsittelystä selkeämpää (Lantz 2013, 56). Aikaisempia esimerkkejä mukaillen ensimmäisen osallistujan tiedot voisi sisällyttää yhteen listaan seuraavanlaisella komennolla: ”osallistuja1 <- list(fullname = osallistujat[1], sukupuoli = sukupuoli[1])”. Kun konsoliin kirjoittaa ”osallistuja1”, se tulostaa osallistujan nimen ja sukupuolen. Listaakaan ei käytetä opinnäytetyössä varsinaisen datan käsittelyyn, vaan erilaisten komentojen sisällyttämiseen samaan listaan kuvaajien visuaalista teemaa määriteltäessä.

Data frame on se datatyyppi, jota ohjelmassa käytetään varsinaisen piirrettävän datan käsittelyyn. Data framessa yhdistyy listojen ja vektorien ominaisuudet, ja se on rakenteeltaan kuin taulukko (Lantz 2013, 58). Edellisten kappaleiden esimerkeistä voitaisiin koostaa data frame

seuraavanlaisella komennolla:” osallistujadata <- data.frame(osallistujat, sukupuoli, stringsAsFactors = FALSE)”. StringsAsFactors-parametri on sisällytetty komentoon, jotta data.frame-komento ei muuta kaikkien muuttujien datatyyppejä faktoriksi. Komento luo data framen, jota voi RStudiossa tarkastella kuvan 5 kaltaisessa näkymässä.




Kuvio 5 Esimerkin data frame RStudiossa

Vaikka opinnäytetyössä käytettävä data on sisällytetty data frameihin, sitä ei kuitenkaan syötetä manuaalisesti R:llä, kuten esimerkissä. Sen sijaan data haetaan ulkoisesta lähteestä, mihin tarkoitukseen on kehitetty monia hyödyllisiä paketteja ja komentoja. Näistä kuitenkin kerrotaan tarkemmin paketteja ja itse ohjelmaa käsittelevissä luvuissa.

### 4.2.3 Kuvaajan piirtäminen

Itse kuvaajan piirtäminen tapahtuu omalla komennollaan, jossa määritellään käytettävän datan lähde ja kuvaajan visuaaliset ominaisuudet. R sisältää sellaisenaan ”plot()”-komennon, mutta pakettien ansiosta käytössä on lukuisia muita komentoja, joilla kuvaajia voi piirtää. Opinnäytetyössä on käytetty ggplot2-paketin ”ggplot”-komentoa, joten tässä luvussa syvennytään vain sen toimintaan.

Ggplot-komentoa käytettäessä siinä tulee määritellä ensin käytettävä data. Slaidisetissä käytettävän ggplot-komennon alkuosa näyttää suurimmassa osassa alaohjelmia tältä: ”p <- ggplot(data=a_date, aes(x=Date, y=Value, fill= factor(SeriesName, labels = labels)”. Komennossa luodaan ”p”- niminen kuvaaja. Data kuvaajan piirtämiseen haetaan ”a_date”-nimisestä data framesta. Aes()-komento taas määrittelee, miten data framen tietoja käytetään. Esimerkkikomennossa ”a_date”:n sisältämä muuttuja ”Date” asettuu x-akselille, ja ”Value” taas y-akselille. Fill-parametrin avulla taas määritellään, että ”a_date”:n data erotellaan ”SeriesName”-nimisen muuttujan mukaan. Jos ”SeriesName”-muuttuja sisältäisi kahta erilaista arvoa, esimerkiksi ”Suomi” ja ”Ruotsi”, ggplot-komento erottelisi toisistaan Suomea ja Ruotsia koskevat havainnot, ja piirtäisi niiden tiedot kuvaajaan erikseen. Esimerkiksi viivakuvaajaan

piirtyisi kaksi viivaa, toinen Ruotsin ja toinen Suomen tiedoista. Esimerkin sisältämän labels-parametrin toimintaa käsitellään tarkemmin selitteitä eli labeleita käsittelevässä luvussa.

Ggplot-komento toimii niin, että käytettävän datan määrittelyn jälkeen siihen lisätään komentoja ikään kuin tasoina (RDocumentation 2018a). Komentoja voi lisätä +-merkillä toistensa perään, ja niillä voidaan määritellä esimerkiksi se, mitä kuvaajatyyppiä käytetään, mikä on kuvaajan otsikko, ja millaisia värejä kuvaajassa käytetään. Esimerkiksi jos ggplot-komennon jälkeen lisätään komento ”geom_bar()”- piirtäisi komento annetusta datasta pylväskuvaajan. ”Labs()”-komennolla taas voisi määritellä kuvaajan otsikkoa ja kuvatekstiä. Lopputuloksena voi syntyä hyvinkin pitkä ja yksityiskohtainen komento, joten Slaidisetin koodissa ggplot-komennot onkin sisällytetty omiin tiedostoihinsa, joita kierrätetään ohjelmassa.

Kuvaajien yksilöllisiä ominaisuuksia määritellään niin kutsutuilla parametreilla. Parametrit arvoineen lisätään komentoihin argumentteina muodossa ”parametri = arvo” (Rahlf 2014, 45). Esimerkiksi kaikissa Slaidisetin kuvaajien ggplot-komennoissa tulee y-akselissa nollan kohdalla olevan viivan oltava hieman muuta ruudukkoa paksumpi, mikä määritellään ggplot-komennossa näin: ” geom_hline(yintercept = 0, color = "#000000", size = 0.6)”. Komennon ”geom_hline” sisällä annetaan arvot parametreille ”yintercept”, ”color”, ja ”size”. Parametrien arvot voidaan antaa myös aikaisemmin määriteltynä muuttujana. Esimerkiksi jos aikaisemmin koodissa olisi määritelty muuttuja ”black” komennolla ”black <- ”#000000””, voisi esimerkkikoodissa käyttää tätä muuttujaa värikoodin sijaan. Tämän kaltaista ratkaisua onkin Slaidisetin kuvaajissa sovellettu niin, että eri kuvaajat voivat käyttää samaa ggplot-komentoa erilaisista parametrien arvoista huolimatta. Näitä parametrien arvoja sisältäviä muuttujia kutsutaan raportissa ”parametrien muuttujiksi”, eikä niitä tule sekoittaa esimerkiksi data framejen muuttujiin.

Kun mikä tahansa kokonaisen kuvaajan piirtävä komento ajetaan R:llä, avautuu niin sanottu graphics device, jossa piirretty kuvaaja näkyy. RStudiossa valmis kuvaaja näkyy oikeassa alakulmassa ”Plots”-välilehdellä. Jotta kuvaaja kuitenkin saataisiin vielä käytännöllisempään muotoon, esimerkiksi kuvatiedostoksi, tulee se jotenkin tallentaa tähän muotoon. Niin RGui kuin RStudiokin tarjoavat vaihtoehdon tallentaa kuva graphics devicen valikosta, mutta automaatioratkaisussa tallentamisfunktio on parasta sisällyttää itse ohjelmaan. Kuvaajien tallentamiseen tiedostoina on pakettien ansiosta olemassa monta komentoa, joista opinnäytetyön automaatioratkaisu käyttää ggsave-komentoa. Komennossa määritellään kuvatiedoston nimi, sekä itse tallennettava kuvaaja. Aiemmassa luvussa mainittu esimerkkikuvaaja ”p” voitaisiin tallentaa seuraavanlaisella komennolla: ”ggsave("p.png",p)”. Kuvaaja tallentuisi automaattisesti tietokoneen ”Tiedostot”-kansioon, mutta halutessaan tallennuskansion voi määritellä komennossa.

## 4.3 Paketit

Pakettien käyttämisestä ja niiden sisältämistä komennoista on jo jonkin verran kerrottu aiemmissa luvuissa. Luvussa kerrotaan paketeista yleisesti, ja esitellään muutama automaatioprojektissa käytetty paketti toimintoineen. Projektissa on käytetty montaa eri pakettia, joita ei kaikkia esitellä. Luvun tarkoituksena onkin pohjustaa ohjelman tärkeimpien ominaisuuksien toteutukseen käytettyjä paketteja, ja antaa esimerkkejä siitä, mitä kaikkea pakettien komennoilla voi tehdä.

### 4.3.1 Pakettien käyttö ja toiminta

Kun uusi paketti halutaan ottaa käyttöön, tulee se ensin ladata ja asentaa. Tämän voi tehdä suoraan R:n konsolista ”install.packages(””)”-komennolla (Wickham 2018b). Sulkujen sisään annetaan sen paketin nimi, joka halutaan asentaa. Komento lataa ja asentaa paketin CRAN:sta (Comprehensive R Archive Network), joka on R-koodille tarkoitettu verkosto (CRAN.R-project 2015). Asennuksen jälkeen paketin komennot eivät ole vielä käytettävissä, vaan ne on otettava käyttöön ”library()”-komennolla. Tämän jälkeen paketin komentoja voi käyttää kuten mitä tahansa komentoja. Vaikka paketin asennus tarvitsee tehdä vain kerran, käyttöönotto on tehtävä jokaisessa R-istunnossa edellä mainitulla komennolla, jos paketin komentoja haluaa käyttää. Tästä syystä ohjelman alkuun on aina hyvä sisällyttää lista library()-komentoja, joilla ohjelman tarvitsemat paketit saadaan käyttöön.

Paketit eivät koostu pelkästään komennoista, vaan myös ohjeista ja näytedatasta (Wickham 2018a). RStudiolla käytössä olevia paketteja ja niiden ohjeistusta pääsee tarkastelemaan oikean alakulman näkymästä ”Packages”-välilehdeltä.

### 4.3.2 Ohjelmassa käytettäviä paketteja

Aiemmissa luvuissa on jo mainittu ggplot2-paketin sisältämä komento ggplot, jolla Slaidisetin kuvaajat varsinaisesti piirretään. Itse tiedon hakuun tarvitaan kuitenkin omaa pakettiaan, joka on tässä kyseisessä ohjelmassa RODBC. Tämän paketin komentoja voi käyttää datan noutamiseen esimerkiksi SQL-tietokannasta (Lanz 2013, 65). Ohjelmassa RODBC:n komentoja käytetään määriteltäessä ohjelman omaa getdata-funktiota, josta kerrotaan tarkemmin itse ohjelmaa kuvaavissa luvuissa. Komennolla ”dbconnect <-odbcConnect(””)” R voi ottaa ”dbconnect”-nimisen yhteyden suluissa määriteltyyn tietokantaan. Jos tietokanta vaatii salasanaa, tulee myös se määritellä odcbConnect()-komennossa. Tämän jälkeen voidaan käyttää saman paketin sqlQuery()-komentoa datan noutamiseen tietokannasta. Komennossa SQL-komento syötetään sulkujen sisään. Komento luo data framen poimituista tiedoista. Lopuksi yhteyden tietokantaan voi katkaista odbcClose()-komennolla (Lanz 2013, 66).

Vaikka ggplot-komennolla voi luoda käyttövalmista tilastografiikkaa, tarvitsee Slaidisetin kuvaajien visuaalista ilmettä vielä hioa. Tähän tarkoitukseen ohjelmassa käytetään ensinnäkin

gtable-paketin komentoja, joilla voi luoda niin kutsuttuja grobeja. Grobit eli Grid Graphical Objectit ovat ylimääräisiä graafisia taulukkoelementtejä kuvaajassa (RDocumentation 2018b). Ohjelmassa grobia käytetään y-akselin otsikon luomiseen, sillä ggplot-komennossa y-akseli ei asetu halutulla tavalla. Tähän käytetään gtable_add_grob-komentoa, jossa määritellään grobin tyyppi, tässä tapauksessa teksti, sen sisältö, sekä koordinaatit.

Toinen graafisia elementtejä käsittelevä paketti, jota ohjelmassa käytetään, on magick. Magick sisältää monipuolisesti kuvankäsittelyä mahdollistavia komentoja, joilla kuvaajista pystyy tekemään esimerkiksi animaation (CRAN.R-project 2018). Ohjelmassa magickin komentoja käytetään kuitenkin yksinkertaisempaan kuvankäsittelyyn. Valmiiden kuvaajien png-versioihin lisätään Suomen Pankin logo image_composite()-komennolla, jolla voi yhdistää bittikarttakuvia toisiinsa. Komentoon syötetään yhdistettävien kuvien nimet sekä sen kuvan koordinaatit, joka piirretään toisen kuvan päälle.

### 4.4 R:n etuja ja haasteita

Edellisessä luvussa annettiin esimerkkejä erilaisten pakettien toiminnasta, joilla pyrittiin havainnollistamaan R:n monipuolisuutta niin ohjelmointikielenä kuin tilastografiikan välineenä. Monipuolinen laajennettavuus onkin yksi hyvä syy käyttää R:ää (Rahlf 2014, 3). Tässä luvussa käydään lyhyesti läpi joitakin R:n etuja, sekä myös niitä haasteita, joita ohjelmointikielen käyttöön voi liittyä.

Laajennettavuuden ohella R:n toinen merkittävä etu on sen ilmaisuus (Rahlf 2014, 3). R on täysin ilmainen asentaa ja käyttää, ja on tältä osin etulyöntiasemassa moniin muihin datankäsittelyn ja tilastografiikan välineisiin, jotka vaativat maksullisen lisenssin. Ilmaisuudestaan huolimatta R:llä luodut kuvat eivät häviä laadussa maksullisille vaihtoehdoille. Ilman pakettejakin R:llä voi luoda yleisimpiä kuvaajatyyppejä käyttäviä kuvaajia yhdellä komennolla. Kuvaajien pienimpiinkin yksityiskohtiin voi vaikuttaa, ja kuvaajat voi tallentaa bittikartan lisäksi myös vektorigrafiikkana, kuten automaatioprojektissa on tehtykin. (Rahlf 2014, 3).

R:n käytössä on kuitenkin myös haasteensa. Ensinnäkin potentiaaliset käyttäjät saattavat pitää ohjelmointia vaativaa R:ää vaikeasti lähestyttävänä välineenä (Rahlf 2014, 3). Pakettien ansiosta R on ohjelmointia jo hieman osaavalle käyttäjälle melko yksinkertainen käyttää, mutta saattaa olla haasteellinen käyttäjälle, jolla ei ole aikaisempaa ohjelmointitaustaa. Toinen R:ään liittyvä haaste on sen ajoittainen hitaus. Tämä johtuu siitä, että R tulkitsee sille syötetyt komennot vasta, kun ne suoritetaan, mikä kuormittaa tietokoneen prosessoria (Lim & Tjhi 2015, 27-28).

# 5 Ohjelman tehtävä ja rakenne

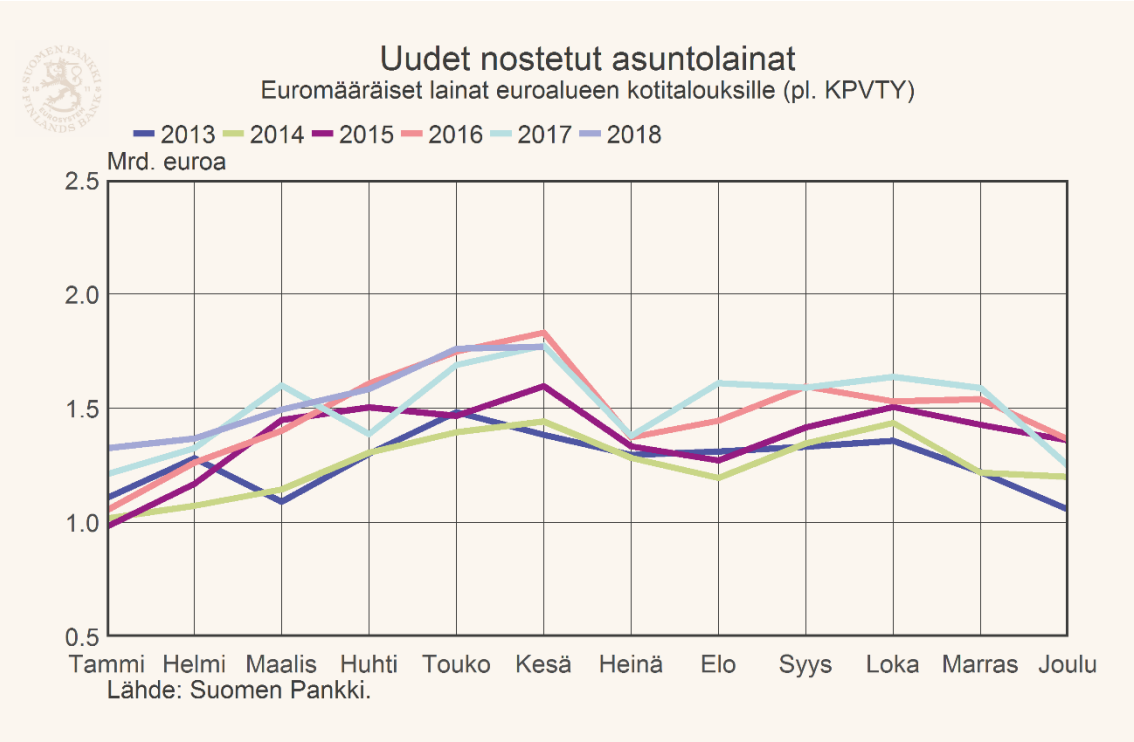
Tässä luvussa kuvataan opinnäytetyönä tuotetun R-ohjelman tehtävää ja rakennetta. Lukujen on tarkoitus esitellä se, mitä ohjelmalla pyritään saamaan aikaan, sekä antaa selkeä yleiskuva ohjelman rakenteesta. Ohjelman tarkempaa toimintaa tarkastellaan myöhemmissä luvuissa.

## 5.1 Ohjelman tehtävä

Ohjelman tehtävä on tuottaa käyttövalmista tilastografiikkaa Suomen Pankin sisäiseen viestintään. Ohjelman tulisi luoda “Slaidisetiksi” kutsuttu esitys kuukausittain päivittyvistä tiedoista. Ohjelmaa tehtäessä prosessi on pyritty automatisoimaan mahdollisimman pitkälle, jotta työaikaa säästyisi.

Koska Slaidisetin sisältö tulee ajan kuluessa mitä luultavimmin muuttumaan, on tärkeää, että ohjelman muokkaaminen ja jatkokehitys on tehty helpoksi. Ohjelman koodi on pyritty kirjoittamaan selkeästi, ja siinä on paljon kommentteja selostamassa ohjelman toimintoja.

## 5.2 Slaidisetin tilastografiikka




Kuvio 6 Viivakuvaaja nostetuista asuntolainoista eri kuukausina vuosittain

Slaidisettiin sisältyy graafisilta ominaisuuksiltaan monen tyyppisiä kuvaajia. Piirrettävän ja verrattavan datan luonteesta riippuen graafiseen esittämiseen on käytetty esimerkiksi viiva-

kuvaajia, vierekkäisiä ja päällekkäisiä pylväskuvaajia, sekä viiva- ja pylväskuvaajien yhdistelmää. Automaatioratkaisun tuottama tilastografiikka pyrkii noudattamaan mahdollisimman uskollisesti ja tarkasti manuaalisen Slaidisetin graafista ilmettä ja ratkaisuja.

Slaidisetin kuvaajat noudattavat pitkälti tilastografiikan hyviä käytäntöjä. Esimerkiksi pylväskuvaajien y-akselin asteikkoa ei ole lyhennetty, ja kuvaajatyyppien valinnassa on otettu huomioon käytettävän datan luonne. Kuvaajat ovat myös Suomen Pankin visuaalisen ilmeen mukaisia, ja käyttävät värejä jotka erottuvat toisistaan selkeästi.

Slaidisetin tilastografiikassa prioriteettina on esittää vertailtavan datan välinen suhde, eikä niinkään keskittyä tarkkoihin lukuihin. Tästä syystä Slaidisettiin sisältyy myös kahden y-akselin kuvaajia, joita ei yleisesti suositella käytettäviksi niiden mahdollisen epäselvyyden vuoksi. Jos vertailtavana kuitenkin on esimerkiksi lainakannan ja korkoprosentin välinen suhde, on kahden y-akselin kuvaaja huomattavasti havainnollistavampi ratkaisu kuin erilliset kuvaajat. Kannan ja koron esittäminen erillisissä kuvaajissa hankaloittaisi tietojen vertailua. Slaidisettiin sisältyvien kahden y-akselien kuvaajien tulkintaa on helpotettu siten, että selitteissä on ilmoitettu, mitataanko dataa vasemmalla vai oikealla akselilla.

Manuaalinen Slaidisetti on pitkälti vakiomuotoinen PowerPoint-esitys, joten myös automatisoidusta ratkaisusta on haluttu koottu esitys. Valmiista esityksestä voi kuitenkin olla hankalaa poimia yksittäisiä kuvaajia käyttöön, joten tilastografiikka tallennetaan myös yksittäisinä png-kuvina. Tämän ansiosta Slaidisetin kuvaajia voidaan esimerkiksi jakaa sosiaalisessa mediassa ja hyödyntää muissa julkaisuissa kätevämmin.

## 5.3 Ohjelman rakenne

Tässä luvussa tarkastellaan ohjelman rakennetta, eli sitä, mistä tiedostoista ohjelma muodostuu, sekä lyhyesti ohjelman toimintaa kokonaisuudessaan. Luvussa käsitellään hieman eri tiedostojen tehtäviä, mutta tarkemmin ohjelman eri osien toimintaa kuvataan seuraavassa luvussa “Ohjelman toiminta”.

Ohjelma koostuu yhdestä pääohjelmasta, useammasta alaohjelmasta, sekä kahdesta erillisestä funktiosta. Kaikki edellä mainitut ovat erillisiä R-tiedostoja, joista vain pääohjelmaa tarvitsee käyttää koko ohjelman ajamiseen. Erillistä käyttöliittymää ei ohjelmaan ole ollut tarve tehdä, sillä ohjelman ajaminen RStudiolla on yksinkertaista, ja sen tulee todennäköisesti suorittamaan R:ää tunteva työntekijä.

Pääohjelma on ikään kuin koko ohjelman runko, ja ohjelman ainut tiedosto, jota Slaidisetin piirtämiseksi ja kokoamiseksi tarvitsee muokata ja ajaa. Pääohjelma on jaettu kahteen osaan: valmistelu ja kuvaajat. Valmisteluosa nimensä mukaisesti valmistelee ohjelman ajamista, ja kuvaajaosa taas sisältää kaikkien kuvaajien koodit.

Valmisteluosa tulisi ajaa erikseen ennen kuvaajaosaa. Valmisteluosassa otetaan käyttöön ohjelman ajamiseen tarvittavat paketit, otetaan käyttöön Suomen Pankin visuaalinen ilme ja getdata-funktio, ja asetetaan työkansio sekä polku kuville. Mikäli kuville ei ole vielä olemassa kansiota, sellainen luodaan. Valmisteluosa sisältää ainoan osan koodia, jota ohjelman normaalin ajamisen yhteydessä tarvitsee muokata. Näitä ovat muuttujat ”kk” ja ”vv”, joihin tulee syöttää senhetkinen kuukausi ja vuosi. Näitä muuttujia käytetään kuvien tallentamiseen ja uuden kansion luomiseen, jos siihen on tarvetta.

Kuvaajaosa muodostuu useasta samankaltaisesta ja peräkkäisestä koodista, joilla kullakin piirretään yksi kuvaaja Slaidisettiin. Nämä kuvaajakoodit koostuvat määriteltävistä parametrien muuttujista, aikasarjahausta, sekä kuvaajan piirtävästä funktiosta, joka määritellään alaohjelmissa. Kuvaajaosan ajaminen kokonaisuudessaan luo kaikista siihen koodatuista kuvaajista png-kuvat ja yhden kootun pdf-tiedoston valmisteluosassa määriteltyyn kansioon. Kuvaajia voi myös ajaa yksitellen, mutta koottua pdf:ää varten on koodi ajettava alusta loppuun.

Pääohjelma tarvitsee toimiakseen alaohjelmia ja kahta funktiota, jotka sijaitsevat omissa erillisissä tiedostoissaan. Näitä koodeja voi käyttää pääohjelmassa viittaamalla niihin, millä on pyritty pitämään pääohjelma kompaktimpana ja välttämään turhaan toistuvaa koodia.

Alaohjelmien tehtävänä on piirtää kuvaaja pääohjelmassa määriteltyjen muuttujien ja annettujen tietojen pohjalta. Se sisältää kuvaajan piirtämiseen käytettävän ggplot-komennon lisäksi myös koodin, joka lisää kuvaajaan Suomen Pankin logon ja tallentaa kuvan png-tiedostona. Alaohjelmiin viitataan pääohjelmassa aina jokaisen erillisen kuvaajakoodin lopussa. Se, mitä alaohjelmaa kunkin kuvaajan piirtämiseen käytetään, riippuu siitä, minkä tyyppinen kuvaaja halutaan piirtää. Viiva- ja pylväskuvaajille, sekä niiden yhdistelmälle on siis kaikille omat alaohjelmansa.

Pääohjelma käyttää ominaisuuksia, jotka on määritelty erillisissä tiedostoissa ”getdata.r” ja ”bof_theme.r”. Getdataan viitataan pääohjelman valmisteluosassa, ja siinä määriteltyä funktiota ”getdata()” käytetään jokaisen kuvaajan koodissa hakemaan kuvaajan tiedot aikasarjahaulla. Bof_theme sisältää koodia Suomen Pankin visuaalisen ilmeen määrittelyyn. Siihen viitataan valmisteluosassa, ja sen luomaa teemaa ”bof_theme_colors” käytetään alaohjelmissa.

# 6 Ohjelman toiminta

Ohjelman varsinaisista toiminnoista on jo kerrottu hieman edellisessä luvussa, mutta tässä luvussa niihin syvennytään tarkemmin. Tarkoituksena on selostaa aiempaa lukua yksityiskohtaisemmin ohjelman toimintoja, ja minkälaisista komennoista ne rakentuvat. Päinvastoin kuin ohjelman rakennetta kuvaavassa luvussa, kerron aluksi getdatan ja bof_themen sisältämien funktioiden toiminnasta, ja siirryn sitten pääohjelman ja alaohjelmien kuvailuun.

## 6.1 Getdata

Getdata-aikasarjahaku on yksi ohjelman tärkeimpiä osia. Se hakee ajankohtaiset tiedot aikasarjatietokannasta aikasarjakoodin avulla. Näin ollen tietoja ei tarvitse päivittää kuvaajiin manuaalisesti, mikä vauhdittaa Slaidisetin kuukausittaista kokoamista huomattavasti. Getdata-funktio määritellään pääohjelmasta erillisessä R-tiedostossa, johon viitataan pääohjelman alussa. Getdata-funktion on kokonaisuudessaan koodannut Juha Itkonen.

Koska getdata-funktion on haettava tiedot ulkoisesta tietokannasta, tarvitsee sen luoda yhteys tällaiseen tietokantaan. Tämä tehdään koodissa luomalla muuttuja ”tsdb_connection”, jolle annetaan arvoksi ”odbcDriverConnect()”-käsky. Käsky sisältää yhteyden muodostamiseen tarvittavat tiedot. Tähän muuttujaan viitataan myöhemmin varsinaisen tietojen hakemisen suorittavassa ”sqlQuery”-komennossa.

Tämän jälkeen koodi määrittelee itse getdata-funktion, joka näyttää tältä: “function(seriesname, frequency = NULL, startDate = NULL, endDate = NULL, validationDate = today(), aggregation = 0, disaggregation = 0)”. Getdataa käytettäessä “seriesname” viittaa aikasarjakoodiin, ja muilla määrittelyillä voidaan seuloa tietoa. Tiedon seulontaa ei kuitenkaan suoriteta pääohjelmassa getdata-funktion avulla, vaan muilla komennoilla.

Funktion määritelmille annetaan tämän jälkeen ehtolausekkeita, joilla määritellään esimerkiksi se, mitä funktio tekee, jos sen käyttäjä ei syötä haun alku- ja loppupäivämäärää. Tässä tapauksessa ehtolausekkeen mukaan funktio hakee koko aikasarjan. Ehtolausekkeilla siis määritellään sitä, miten funktio toimii käyttäjän syöttämien tietojen suhteen.

Tämän jälkeen määritellään ”sqlquery”-muuttuja, jota ei pidä sekoittaa sqlQuery-komentoon. Muuttujalle annetaan arvoiksi SQL-komento, jossa käytetään getdata-funktioon syötettäviä arvoja, kuten aikasarjakoodia ja päivämääriä. Sqlquery-muuttujaan siis luodaan kokonainen SQL-komento yhdistelemällä komennon osia muuttujiin, joiden arvot tulevat getdata-komennon käyttäjän syöttämänä.

Tiedot haetaan tämän jälkeen sqlQuery-komennolla, joka suorittaa halutun SQL-kyselyn tietokantaan. Komennossa käytetään aiemmin luotuja muuttujia tsdb_connection ja sqlquery. Tämän jälkeen määritellään vielä, mitkä tiedot aikasarjatietokannasta halutaan poimia, ja missä muodossa. Getdata kerää aikasarjan tunnuksen, nimen, havaintojen päivämäärät sekä arvot.

Getdata-koodissa määritellään myös ”getmetadata”-funktio, jolla voidaan hakea aikasarjan metatiedot. Ohjelmassa tätä ominaisuutta ei kuitenkaan käytetä, joten sitä ei tarkemmin kuvailla.

## 6.2 Bof_theme

Bof_theme määrittelee kuvaajien visuaalisen ilmeen, ja siihen viitataan kaikissa alaohjelmissa. Bof_themen koodissa listataan ensin muuttujina Suomen Pankin visuaaliseen ilmeeseen kuuluvat värit ja annetaan niille rgb-arvo, eli niiden tarkka väri. Värilistasta tehdään vektori, ja sen jälkeen ggplot-elementti ”list(scale_color_manual(), scale_fill_manual())”-komennolla. Tämän jälkeen koodi luo itse bof_theme-teeman ”theme”-komennolla. Teemassa määritellään kuvaajien ulkoasua: akseleita, ruudukkoa, taustaväriä, legendiä, otsikoita ja kuvatekstiä. Teema määrittelee näiden osien värejä, kokoa, paksuutta, sijaintia ja marginaaleja. Esimerkiksi taustavärin määrittely tehdään tällaisella komennolla: ”plot.background = element_rect(colour = hiekka, fill = hiekka)”. Lopuksi koodi luo värilistasta ja teemasta yhteisen listan “bof_theme_colors”, johon alaohjelmat viittaavat piirtäessään kuvaajia.

## 6.3 Alaohjelmat

Alaohjelmia käytetään kuvaajien varsinaiseen piirtämiseen. Alaohjelmien tarkoituksena on kierrättää tiettyjä koodin osia, jotta niitä ei tarvitsisi koodata useaan kertaan pääohjelmaan. Alaohjelmia on käytössä tällä hetkellä 8 erilaista, joiden erot johtuvat piirrettävien kuvaajien tyypistä sekä muista graafisista ominaisuuksista. Tässä luvussa käydään läpi alaohjelmien toimintaa, sekä miten pääohjelmassa määriteltävien parametrien muuttujien avulla samaa alaohjelmaa käyttämällä saadaan piirrettyä erilaisia kuvaajia.

Alaohjelmat ovat rakenteeltaan samanlaisia ja sisältävät samat toiminnot: kuvaajan piirtävän ggplot-komennon, graafisia elementtejä lisäävän textGrobin, logon lisäämisen, tallentamisen png:nä, sekä kuvaajan lisäämisen graphs-listaan pdf:n tallentamista varten. Ggplot-komennon sisältö onkin ainut asia, joka erottaa alaohjelmat toisistaan. Niin käytettävät data framen muuttujat kuin piirrettävän kuvaajan tyyppi vaihtelevat alaohjelmasta toiseen. Muuttujien toimintaa alaohjelmassa käsitellään tarkemmin omassa luvussaan.

Alaohjelmien alussa sisällytetään ”p”-muuttujaan kokonainen ggplot-komento. Tämä komento saa aina yksilölliset, kuvaajakohtaiset ominaisuudet pääohjelmassa määritellyistä parametrien ja data framejen muuttujista. Näiden kautta ggplot-komentoon siis määritellään pääohjelmassa kuvaajassa käytettävä data, akseleiden ominaisuuksia, labeleita, sekä muita graafisia ominaisuuksia. Ggplot-komennon ensimmäinen osa näyttää suurimmassa osassa alaohjelmia tältä: ” p<-ggplot(data=a_date, aes(x=Date, y=Value, fill= factor(SeriesName, labels = labels)))”. Komennossa data haetaan ”a_date”-nimisestä data framesta, joka on luotu pääohjelmassa. Tämän jälkeen annetaan ne muuttujat, joita x- ja y-akseleiden tulee käyttää. Komennossa määritellään myös muuttuja, jonka mukaan data erotellaan toisistaan, eli ”SeriesName”. Komennossa määritellään myös selitteet, jotka tulevat tässä tapauksessa ”labels”-muuttujasta. Myös selitteiden toimintaa kuvaillaan tarkemmin omassa luvussaan.

Ggplot-komento sisältää komentoja, joilla määritellään kuvaajan graafisia ominaisuuksia, joita kaikkia ei käydä läpi tässä. Aiemmin luotu bof_theme_colors lisätään ggplot-komentoon. Muilla graafisia ominaisuuksia muokkaavilla komennoilla määritellään kuvaajan värejä, viivojen paksuuksia, otsikoita, kuvatekstejä ja akseleiden ominaisuuksia. Esimerkiksi y-akselin minimi, maksimi ja välit määritellään komennossa ” scale_y_continuous(limits = c(y_min, y_max), breaks = seq(y_min, y_max, by = by), expand=c(0,0))”. Muuttujien ”y_min”, ”y_max” ja ”by” arvot määritellään pääohjelmassa kullekin kuvaajalle erikseen. Parametrien muuttujien toiminnasta kerrotaan tarkemmin pääohjelmaa käsittelevässä luvussa.

Vaikka ggplot-komennolla saadaankin piirrettyä varsin hyvää tilastografiikkaa, se ei yksinään riitä luomaan Suomen Pankin visuaalisen ilmeen mukaisia kuvaajia. Kuvaajien piirtämiseen käytetäänkin alaohjelmissa ggplotin lisäksi komentoja, joilla lisätään lopulliseen tilastografiikkaan graafisia elementtejä. Ggplot asettaa akseleiden otsikot automaattisesti y-akselin sivulle ja x-akselin alle. Slaidisetin kuvaajissa y-akselin otsikko halutaan kuitenkin y-akselin kärkeen, kuvaajan yläkulmaan. X-akselin otsikko ei ole lainkaan tarpeellinen, sillä siinä käytettävät vuosiluvut ja kuukausien nimet ovat tarpeeksi yksiselitteisiä. Molempien akseleiden otsikot piilotetaan ggplot-komennossa komennolla ”theme(axis.title.x = element_blank(), axis.title.y = element_blank())”. Koska y-akselille kuitenkin halutaan otsikko, se lisätään ggplot-komennon jälkeen TextGrob-elementtinä. Ensin ”p”-kuvaajasta tehdään grob-elementti ”g” komennolla ”g = ggplotGrob(p)”. Tämän jälkeen lisätään y-akselille otsikko. Komento näyttää tältä: ”g = gtable_add_grob(g, textGrob(ytitle, x=1, hjust=0, vjust=-0.1, gp = gpar(col=harmaanmusta, fontsize=16)), t=5, l=3, clip = "off")”. Kuvaajaan lisätään siis komennossa määritettyihin koordinaatteihin ”ytitle”-muuttujaan pääohjelmassa määritelty y-akselin otsikko, esimerkiksi ”Mrd. euroa” tai ”%”.

Tämän jälkeen kuva tallennetaan ensimmäisen kerran ggsave-komennolla, joka näyttää tältä: ”ggsave(paste0(slaidipolku,"Kuvio_",gsub(" ","_",kuvio,fixed=TRUE),".png"), g, width = 25.85, height = 16.89, units = "cm")”. Komennossa määritellään kuvan tallennuspolku ja kuvanimi, jotka muodostuvat vakituisista teksteistä kuten ”Kuvio_” ja ”.png”, sekä muuttujista ”slaidipolku” ja ”kuvio”, joiden sisältö määritellään pääohjelmassa. ”Slaidipolku” määritellään pääohjelman valmisteluosassa joten se on kaikille kuvaajille yhteinen, mutta ”kuvio” määritellään jokaiselle kuvaajalle erikseen. Jos ”slaidipolku” olisi esimerkiksi ”C:\Users\User\Documents”, ja ”kuvio” ”7 Yleisön lainakannan kasvu”, tallentaisi komento ”Kuvio_7_Yleisön_lainakannan_kasvu.png”-nimisen kuvan polussa määriteltyyn kansioon.

Kuvaaja tallennetaan kertaalleen png-tiedostona eli bittikarttana, jotta siihen saadaan lisättyä Suomen Pankin logo. Tämän jälkeen ”kuvanimi” muuttujan arvoksi annetaan äsken tallennetun kuvan koko polku samaan tapaan kuin kuvaa tallennettaessa. Muuttujalle ”logo” taas annetaan arvoksi R-tiedostojen kanssa samassa kansiossa sijaitseva Suomen Pankin logo ko-

mennolla ”logo <- image_read("SPlogo_keskikoko.png")”. Sama tehdään myös äsken tallennetulle kuvaajalle komennolla ”graph_bg <- image_read(kuvanimi)”. Tämän jälkeen kuvat yhdistetään ja yhdistetty kuva tallennetaan logottoman png-kuvan päälle. Yhdistämiseen käytettävä koodi näyttää tältä: ”logoplot <- image_composite(graph_bg, logo, offset = "+45+106")”. Offset määrittelee sen, mitkä ovat logon koordinaatit. Valmis kuvaaja tallennetaan ggsaven sijaan image_write komennolla: ”image_write(logoplot, kuvanimi)”.

Alaohjelman viimeinen toiminto on lisätä piirretty kuvaaja graphs-listaan, jonka avulla kaikki Slaidisetin kuvaajat saadaan kerättyä koottuun pdf-tiedostoon. Graphs-listan toiminnasta kerrotaan tarkemmin ratkaisuja käsittelevässä luvussa.

### 6.4 Pääohjelma

Kuten aiemmissa luvuissa on mainittu, pääohjelma koostuu kahdesta osasta. Ensimmäinen näistä on valmisteluosa, jonka toimintaa on jo hieman selostettu. Aivan ensimmäinen asia, joka valmisteluosassa tehdään, on pankin proxy-palvelimen tallentaminen R:n asetuksiin muokkaamalla Renviron-tiedostoa. Tiedostoon lisätään proxy-palvelimen osoitteet. Tämä vaihe tarvitsee suorittaa vain kerran, joten komennot ovat kommenttien sisällä.

Seuraavaksi haetaan pääohjelman käyttämät paketit library()-komennolla. Komento ottaa käyttöön asennetut paketit, ja ilmoittaa, jos pakettia ei ole asennettu. Koodi sisältää myös ohjeen pakettien asennukseen, joka tehdään install.packages()-komennolla. Paketit tarvitsee asentaa vain kerran, mutta käyttöönotto on tehtävä aina kun RStudion on sulkenut ja avannut uudestaan.

Tämän jälkeen asetetaan työkansio setwd()-komennolla. Tätä komentoa ei yleensä suositella käytettäväksi, sillä se vaikeuttaa koodin jakamista muiden käyttöön. Koska ohjelmaa ei kuitenkaan ole suunniteltu jaettavaksi, komento ajaa hyvin tehtävänsä tässä kyseisessä ohjelmassa. Slaidisetin työkansio sisältää kaikki tarvittavat koodit ja muut materiaalit, sekä alakansion, jonne valmiit kuvaajat tallentuvat.

Työkansion lisäksi määritellään myös polku, jonne valmiit kuvaajat ja koottu pdf tallentuvat. Ensin luodaan muuttuja “pvm”, jonka tulisi sisältää senhetkinen kuukausi ja vuosiluku. Muuttuja ”pvm” muodostetaan kahdesta muusta muuttujasta, ”kk” ja ”vv”, joille ohjelman ajaja asettaa oikeat arvot, esimerkiksi ”02” ja ”2019”. Muuttujan ”pvm”-luova komento näyttää tältä: ”pvm = paste0(kk,"-",vv)”. Tätä muuttujaa käytetään uuden muuttujan “slaidipolku” luomiseen, sekä myös pdf:n nimeämiseen. Kuvien tallentumispaikan määrittävä “slaidipolku” luodaan seuraavanlaisella komennolla: “slaidipolku <- paste0("X:/RMTI/RMTITA/Slaidisetti/Slaidit/",pvm,"/")”. Paste()-komento siis yhdistää vakiona pysyvän polun aina kuukausittain muuttuvaan “pvm”-muuttujaan, jotta kuvat tallentuisivat joka kuukausi eri kansioon. Samanlaisella logiikalla toimii myös pdf:n tallennus aivan ohjelman lopussa.

Määritelty polku tarkistetaan ja luodaan vielä ehtolauseella. Tarkistus tehdään varmistamaan se, etteivät uudet kuvaajat tallennu vahingossa vanhojen kuvaajien päälle, jos käyttäjä on unohtanut muuttaa “kk” ja ”vv” -muuttujat. Jos polun mukainen kansio on jo olemassa, antaa ohjelma virheilmoituksen. Jos taas ei, polun mukainen uusi kansio luodaan. Jos käyttäjän taas on nimenomaan tarkoitus muokata olemassa olevan kansion sisältöä, koodin kommentit neuvovat jättämään tämän vaiheen väliin.

Aiemmin käsiteltyjen getdatan ja bof_themen sisältämät toiminnot ja ominaisuudet otetaan käyttöön viittaamalla niihin source()-komennolla. Ajamalla nämä komennot R ajaa näiden tiedostojen koodit ilman, että käyttäjän tarvitsee erikseen avata tiedostoja. Ohjelma voi tämän jälkeen vapaasti käyttää näiden koodien luomia listoja ja funktiota ilman, että source()-komentoa tarvitsee käyttää uudelleen saman istunnon aikana.

Seuraavaksi ajetaan pääohjelman toinen osa, kuvaajaosa. Ennen varsinaisten kuvaajien koodien ajamista luodaan lista “graphs”, jota käytetään pdf:n kokoamiseen ja tallentamiseen. Tämän listan toimintaa kuvataan myöhemmin lisää.

Kuvaajakoodit ovat rakenteeltaan samankaltaisia. Ne kaikki koostuvat pääosin erilaisten parametrien muuttujien määrittelyistä, tietojen hausta sekä muokkaamisesta, sekä lopuksi alaohjelman ajamisesta. Joillain kuvaajilla, kuten esimerkiksi kahden Y-akselin kuvaajilla, on enemmän parametreja kuin toisilla, ja myös tiedon muokkaamistarve voi vaihdella kuvaajien välillä. Kuten aiemmin on mainittu, ajettava alaohjelma riippuu siitä, millainen kuvaaja tiedoista halutaan piirtää.

Jokaisen kuvakoodin alussa ilmoitetaan kommentissa kuvaajan numero sekä nimi koodin selkeyttämiseksi. Tämän jälkeen määritellään arvoja listalle parametrien muuttujia, joita käytetään alaohjelmassa. Nämä muuttujat määrittävät kaikissa kuvaajakoodeissa aikasarjan loppu- ja alkupäivämäärän, y-akselin minimi- ja maksimiarvot sekä välit, x-akselin aikaformaatin, viivakuvaajan viivan paksuuden, kuvanimen, otsikon, alaotsikon, y-akselin otsikon, kuvatekstin, sekä selitteet. Lisäksi kahden y-akselin kuvaajien koodissa on parametrien muuttujia oikean y-akselin säätöön, mutta niitä kuvaillaan tarkemmin ratkaisuja käsittelevässä luvussa.

Tiedot, joista kuvaaja piirretään, haetaan getdata-komennolla. Komento on sijoitettu data.frame()-komennon sisään, jotta tiedot saataisiin helposti tarkasteltavaan ja muokattavaan muotoon. Yhden aikasarjahaun komento kuvaajakoodissa näyttää siis tältä: ”df1<-data.frame(getdata("AIKASARJAKOODI"))”. Koodi luo “df1”-nimisen data framen, johon sisältyy koodin mukaisen aikasarjan tiedot. Koska kuvaajiin useimmiten käytetään tietoja useammasta aikasarjasta, luodaan useammalla aikasarjahaulla useampi data frame, esimerkiksi “df2” ja “df3”, ja yhdistetään nämä rbind()-komennolla. Esimerkiksi koodi “total<-rbind(df1,df2,df3)” luo “total”-nimisen data framen, jossa on kaikkien yhdistettyjen data framejen tiedot.

Joidenkin kuvaajien tapauksessa aikasarjojen tiedoilla on tehtävä laskutoimituksia. Esimerkiksi lainojen ja talletusten tiedoista tulee laskea kokonaismarginaali vähentämällä lainat talletuksista. Tämän kaltainen laskutoimitus on toteutettu ohjelmassa seuraavanlaisesti: “df23<-data.frame(df2$Value - df3$Value)”. Koodi luo uuden data framen “df23”, jonka muuttujan arvo on koodissa käytettyjen muuttujien erotus. Tämän jälkeen uuden data framen rakennetta muutetaan niin, että sen voi yhdistää muiden data framejen kanssa. Yhdistettävien data framejen muuttujien on oltava samat, joten ne lisätään uuteen data frameen, esimerkiksi “df23$DatasetID = "001"” ja ”df23$SeriesName = "S_Marg"”. ”Period”-muuttujuan arvot taas voidaan kopioida toisesta data framesta, esimerkiksi näin: ”df23$Period<-df2$Period”.

Ennen kuin tietojen pohjalta piirretään alaohjelmalla kuvaaja, tulee niitä vielä hieman muokata. Tietojen tulee olla pitkässä formaatissa, jotta ne piirtyisivät oikein. Tämä tehdään seuraavanlaisilla komennoilla:”colnames(total) [colnames(total)=="Period"]<-"Date"” ja ”a_long<-subset(total, select = c(Date, SeriesName, Value))”. ”Total”-data framen pohjalta luodaan uusi data frame “a_long”. ”Date”-muuttujan päivämäärä muutetaan "%d.%m.%Y"-muotoon, eli päivä-kuukausi-vuosi-järjestykseen ”a_long$Date<-as.Date(a_long$Date, format="%d.%m.%Y")”-komennolla. Tämän jälkeen a_longin pohjalta luodaan taas uusi data frame “a_date”, joka suodattaa a_longin tietoja aiemmin kuvaajakoodissa muuttujiin määriteltyjen alku- ja loppupäivämäärien perusteella. Komento näyttää tältä: ”a_date<-subset(a_long, Date>=start1 & Date<=end1)”. Tähän data frameen sisällytetään siis kaikki kuvaajaan piirrettävä tieto oikeassa muodossa ja oikealta ajalta, sillä sitä käytetään kuvaajakoodin lopussa ajettavassa alaohjelmassa. Ohjelma sisältää kuitenkin tällä hetkellä muutaman kuvaajan, joka käyttää poikkeuksellisesti useampaa data framea. Näistä kuvaajista kerrotaan tarkemmin ratkaisuja käsittelevässä luvussa.

Kun parametrien muuttujien arvot on määritelty ja käytettävä data on saatu haluttuun muotoon, ohjelma viittaa alaohjelmaan ja ajaa siinä määritellyn ”p”-funktion. Komento näyttää tältä:”source("Geom_line_longv4.R")”. Esimerkissä on käytetty alaohjelmaa, joka piirtää yksinkertaisen viivakuvaajan. Jos samoista tiedoista haluttaisiinkin piirtää pylväskuvaaja, tulisi source()-komentoon syöttää toisen alaohjelman nimi. Alaohjelmien toiminnasta on kerrottu tarkemmin aiemmassa luvussa.

Alaohjelman lopussa piirretty kuvaaja lisätään aiemmin mainittuun graphs-listaan. Pääohjelman lopussa listaan kertyneet kuvaajat kootaan niin sanotuksi multiplotiksi “marrangeGrob”-komennolla, ja tallennetaan pdf:nä ggsave-komennolla. Kuten “slaidipolkuun” myös kootun pdf:n nimeämiseen käytetään “pvm”-muuttujaa.

### 6.4.1 Parametrien muuttujat

Jotta jokaiselle kuvaajalle ei tarvitse koodata omaa alaohjelmaansa ja ggplot-komentoa, tehdään kuvaajien yksilöllisten ominaisuuksien määrittely pääohjelmassa. Tähän käytetään niin

kutsuttuja parametrien muuttujia, joille voi asettaa haluamansa arvon. Esimerkiksi alaohjelmat käyttävät muuttujia “y_min” ja “y_max” määrittämään y-akselin asteikon komennolla: “scale_y_continuous(limits = c(y_min, y_max)”. Näille muuttujille annetaan pääohjelmassa jokaisen kuvaajan kohdalla arvo, esimerkiksi “y_max=30” ja “y_min=0”. Kun alaohjelma ajetaan, se piirtää ruudukon, jossa y-akseli alkaa nollasta ja päättyy kolmeenkymmeneen. Kun näille muuttujille annetaan uudet arvot seuraavaa kuvaajaa piirrettäessä, vanhat arvot korvaantuvat. Näin samaa alaohjelmaa voidaan kierrättää ilman, että sitä itseään tarvitsee mitenkään muokata eri kuvaajien yksilöllisten määritysten mukaan. Tässä luvussa tarkastellaan pääohjelmassa määriteltävien ja alaohjelmassa käytettävien parametrien muuttujien toimintaa.

Ensin kuvaajakohtaisissa koodeissa määritellään poiminta-ajanjakso “start1” ja “end1” -muuttujilla. Näiden muuttujien pohjalta ei pelkästään määritellä kuvaajan graafisia ominaisuuksia, tässä tapauksessa x-akselia, vaan myös aikaväliä, jolta kuvaajaan poimitaan tietoa. X-akselin määrittämiseen käytetään alaohjelmien ggplot-komennoissa komentoa “scale_x_date(limits=c(as.Date(start1),as.Date(end1))”. X-akselilla käytetään Slaidisetin kuvaajissa lähes poikkeuksetta vuosilukua, joten “start1”:n ja “end1”:n arvot on määritelty päivämääriksi. Pääohjelmassa “start1” on annettu sellaisenaan kullekin kuvaajalle ilman muuttujia, esimerkiksi “start1 = "2009-01-01"”. Muuttuja “end1” sen sijaan muodostuu muuttujasta “vv” ja päivämäärästä: “end1 = paste0(vv,"-12-31")”. Muuttuja “vv” merkitsee vuosilukua, ja käyttäjän on määriteltävä se ohjelman valmisteluosiossa. Näin vuosilukua ei tarvitse vaihtaa vuoden vaihtuessa jokaiseen ohjelman kuvaajakoodiin.

Muuttujien “y_max” ja “y_min” toiminnasta mainittiin hieman luvun alun esimerkissä. Y-akselin määrittelyyn liittyy näiden muuttujien lisäksi “by”-muuttuja, jolla määritellään asteikon välit. Esimerkiksi 0-10 asteikkoa ei välttämättä tarvitse esittää kymmenen numeron asteikolla, vaan siihen voi käyttää myös viiden numeron asteikkoa, jossa näkyy vain joka toinen luku. Tällainen asteikko saataisiin aikaan seuraavilla määrityksillä: “y_max=10, y_min=0, by = 2”. Alaohjelmassa muuttujia hyödyntävä komento näyttää tältä: “scale_y_continuous(limits = c(y_min, y_max), breaks = seq(y_min, y_max, by = by), expand=c(0,0))”. Asteikkoa voi muokata hyvin monipuolisesti, ja esimerkiksi negatiiviset ja desimaaliluvut kelpaavat asteikolle.

Y-akselin määrittämisen jälkeen kuvaajakoodissa voidaan muokata X-akselin formaattia “xaxis”-muuttujalla. Pääsääntöisesti X-akselin arvot ovat joko vuosia tai kuukausia, ja muuttujalle annetaan arvoksi joko “%Y” tai “%b” riippuen siitä, kumpaa formaattia X-akselilla halutaan käyttää. Alaohjelmissa muuttujaa käytetään aikaisemmin mainitun “scale_x_date()”-komennon sisällä “breaks”-komennossa: “breaks=seq(as.Date(start1), as.Date(end1), "years"), date_labels =paste(c(rep(" ",11), xaxis), collapse=""),  expand=c(0,0)”.

Suuri osa Slaidisetin kuvaajista on viivakuvaajia. Vaikka kuvaajien viivan paksuus onkin suurimmassa osassa kuvaajista sama, voi joidenkin kuvaajien tapauksessa tulla tarpeeseen vaihtaa viivan paksuutta. Näin on hyvä tehdä esimerkiksi selkeyttämään kuvaajaa silloin, kun kuvaaja sisältää dataa tavallista pidemmältä aikaväliltä, ja on piirretty tiiviimmin. Viivan paksuutta voidaan muuttaa “linesize”-muuttujalla, joka toimii alaohjelmissa seuraavanlaisesti: “geom_line(size = linesize)”.

Kuvaajan kuvanimi määritellään myös muuttujan avulla. Muuttujalle “kuvio” annetaan arvoksi kuvaajan numero ja nimi, esimerkiksi “kuvio = "19 Opintolainat"”. Alaohjelmissa muuttujaa käytetään ggsave-komennossa, jossa kuvaaja tallennetaan png:nä seuraavanlaisesti: “ggsave(paste0(slaidipolku,"Kuvio_",gsub(" ","_",kuvio,fixed=TRUE),".png"), g, width = 25.85, height = 16.89, units = "cm")”. Komennossa yhdistetään ensin kaikille kuvaajille yhteinen “slaidipolku”-muuttuja yksilöllisen “kuvio”-muuttujan kanssa, jotta kuvaajalle saadaan kokonainen tallennuspolku. Tämän jälkeen määritellään tallennettava kuvaaja (g), sekä kuvan mitat.

Tilastografiikka tarvitsee tuekseen otsikoita ja selitteitä, jotka helpottavat datan ymmärtämistä ja asettavat sen oikeaan kontekstiin. Kuvaajakoodiin sisältyy luonnollisesti muuttujat, joilla muokata kuvaajan otsikkoa, alaotsikkoa ja lähdetekstiä. Näitä kolmea käytetään alaohjelmissa “labs”-komennossa: “labs(title=title, caption=caption, subtitle=subtitle)”. Muuttujat ovat nimeltään samoja, kuin “labs”-komennon parametrit.

Myös akseleiden otsikot vaihtelevat kuvaajasta riippuen, joten niiden muokkaamiseen on hyvä olla muuttujat. Y-akselin otsikkoa muokataan “ytitle”-muuttujalla, ja x-akselia “xtitle”-muuttujalla. X-akselin otsikkoa ei kuitenkaan käytännössä katsoen käytetä lainkaan, sillä akselilla käytettävät vuosiluvut tai kuukaudet ovat tarpeeksi yksiselitteisiä. Koska ggplot piirtää y-akselin otsikon automaattisesti akselin sivulle, eikä Suomen Pankin graafisen ilmeen mukaisesti akselin päähän, käytetään y-akselin lisäämiseen alaohjelmassa erillisen graafisen elementin komentoa “gtable_add_text_grob()”. Komento näyttää kokonaisuudessaan tältä: “g = gtable_add_grob(g, textGrob(ytitle, x=1, hjust=0, vjust=-0.1, gp = gpar(col=harmaanmusta, fontsize=16)), t=6.5, l=4, clip = "off")”.

Myös labeleiden eli selitteiden määrittelyyn käytetään muuttujia, mutta niiden toimintaa kuvataan tarkemmin ratkaisuja käsittelevässä luvussa.

### 6.4.2 Data framen muuttujat

Alaohjelmat hyödyntävät kuvia piirtäessään parametrien muuttujien lisäksi myös data framejen muuttujia. Muuttujat tulevat suoraan data frameista, ja niitä käytetäänkin määrittämään sitä, mitä dataa ohjelman halutaan piirtävän. Muuttujat ovat kaikissa alaohjelmissa samat, muutamaa poikkeusta lukuun ottamatta. Tässä luvussa muuttujilla viitataan nimenomaan data

framen muuttujiin, ei pääohjelmassa määriteltäviin parametrien muuttujiin tai muihin muuttujiin, joita ohjelma sisältää.

Aikasarjoista haetuista tiedoista kootuissa data frameissa on neljä eri muuttujaa: ”DatasetID”, ”SeriesName”, ”Date” ja ”Value”. ”DatasetID” sisältää aikasarjan datasetin tunnuksen. Tätä ei käytetä ohjelmassa. ”SeriesName” taas on aikasarjan kokonainen nimi eli aikasarjakoodi, joka on syötetty getdata()-komentoon tietojen hakemiseksi. SeriesNamea käytetään erottelemaan eri aikasarjoista haetut tiedot kuvaajaa piirrettäessä, jos tiedot on koottu samaan data frameen. Date viittaa päivämäärään, jolloin aikasarjan havainto on kirjattu. Kuten aiemmin on kerrottu, aikamäärettä käytetään x-akselilla. Value taas on itse havainto, joka aikasarjaan on kirjattu, kuten tietty kanta tai korkoprosentti. Value sijoitetaan y-akselille.

Muuttujia käytetään alaohjelmien ggplot-komennossa, jonka toiminnasta on kerrottu tarkemmin alaohjelmia käsittelevässä luvussa. Komennossa mainitaan ensin käytettävä data frame, ja sen jälkeen niiden muuttujien arvot, jotka halutaan kullekin akselille, sekä muuttuja, jonka mukaan tiedot erotellaan. Jos esimerkiksi halutaan piirtää kuvaaja, jossa vertaillaan Suomen ja Ruotsin asuntolainojen vuosikasvua, tarvitaan ensinnäkin data frame, johon on pääohjelmassa yhdistetty Suomen ja Ruotsin tiedot eri aikasarjoista. Y-akselilla käytetty muuttuja tulee olla Value ja x-akselilla taas Date. Tietoja erotteleva muuttuja on SeriesName, sillä Suomen ja Ruotsin tiedot on alun perin haettu eri aikasarjoista, eli niillä on eri aikasarjakoodit.

# 7 Ratkaisut

## 7.1 Logot

Logon lisääminen kuvaajaan on yksi niistä ominaisuuksista, jonka idea sai alkunsa verkkojulkaisukokouksesta. Logoa toivottiin kuviin, sillä se on osa Suomen Pankin PowerPoint-teemaa, ja näin ollen esillä myös manuaalisesti koottavassa Slaidisetissä, joka toimii automatisoidun Slaidisetin mallina.

Logon lisääminen tehdään käyttäen magick-paketin funktioita. Magickin toiminnot keskittyvät kuvankäsittelyyn, ja pystyvät paikoin yllättävän monipuoliseen kuvanmuokkaukseen. Ohjelmassa käytetään magickin funktioita kuvien tallentamiseen ja yhdistämiseen.

Koodi logon lisäämiseksi on sisällytetty jokaiseen alaohjelmaan, ja se on sama kaikissa alaohjelmissa. Logon lisääminen tapahtuu sen jälkeen, kun kuvaaja on piirretty ja tallennettu ensimmäisen kerran png:nä. Kuvalle luodaan objekti “kuvanimi” seuraavanlaisella komennolla: “kuvanimi<-paste0(slaidipolku,"Kuvio_",gsub(" ","_",kuvio,fixed=TRUE),".png")”. Kuvanimeen käytetään siis pääohjelmassa määriteltyä “kuvio”-muuttujaa, esimerkiksi “kuvio = "7 Kuvan nimi tähän"”. Tällä muuttujalla ja komennolla kuvan nimeksi tulisi siis: “Kuvio_7_Ku-

van_nimi_tähän.png”. Samanlaista komentoa käytetään myös kuvan tallentamiseen ensimmäisen kerran, vain yhdistettynä ggsave()-funktioon. Kuvan oikea nimi siis on sama, kuin “kuvanimi”.

Kun kuvanimi on luotu, tuodaan ohjelmaan Suomen Pankin logo. Logo on png-tiedosto, joka sijaitsee samassa kansiossa kuin ohjelma. Logon voi siis lisätä yksinkertaisesti magickin image_read()-komennolla näin: “logo <- image_read("SPlogo_keskikoko.png")“. Myös tallennettu kuvaaja lisätään samalla tavalla: “graph_bg <- image_read(kuvanimi)”. Tämän jälkeen kuvat yhdistetään image_composite()-komennolla: “logoplot <- image_composite(graph_bg, logo, offset = "+45+106")”. Komennossa ikään kuin taustalle jäävä kuva on mainittava ensin, ja sen päälle lisättävä kuva sen jälkeen. “Offset” määrittää päälle lisättävän kuvan sijainnin alemman kuvan päällä. Näillä koordinaateilla logo sijoittuu kuvaajan vasempaan yläkulmaan. Lopuksi yhdistetty kuva “logoplot” tallennetaan image_write()-komennolla: “image_write(logoplot, kuvanimi)”. Yhdistetty kuva tallentuu siis aiemmin tallennetun, logottoman kuvan päälle.

Vaikka logon lisääminen vaikuttaa viimeistellyssä lopputuotoksessa yksinkertaiselta toiminnolta, sen kehittäminen on ollut yksi projektin haastavimmista osuuksista, jonka kanssa on jouduttu myös tekemään kompromisseja. Logon lisäämiseen on testattu useampaa pakettia funktioineen, ja tämänhetkisen ratkaisun hiominen on jatkunut aina projektin loppumetreille saakka.

Lähestyin logon lisäämistä kuvaajaan kahdesta eri näkökulmasta. Ensimmäinen idea oli säätää kuvaajalle taustakuva, jossa logo olisi sijoitettu vasempaan yläkulmaan. Toinen idea taas oli asettaa logo pienenä kuvatiedostona piirrettyyn kuvaajaan. Molempia ideoita testattiin erilaisilla paketeilla, ja lopulta jälkimmäinen osoittautui helpommin toteutettavaksi.

Taustakuvan lisäämistä testattiin ggimage-paketin ggbackground()-komennolla. Komennon tulisi yhdistää kuvatiedostona löytyvä taustakuva valmiin kuvaajan kanssa. Komento itsessään toimi testattaessa, mutta itse taustakuvan sijoittaminen oikein osoittautui hankalaksi. Kuvasuhde oli usein vääränlainen, ja logon sijainti oli myös haasteellinen määrittää. Koin taustakuvaan verrattuna paremmaksi vaihtoehdoksi lisätä logo erikseen niin, että sen koordinaatteja pystyisi säätämään suoraan R:llä. Taustakuvan lisäämistä kokeiltiin myöhemmin vielä pdf-tiedostoon, mutta taustakuva ei testatessa piirtynyt textGrobin kanssa, joka on huomattavasti tärkeämpi osa kuvaajia. Taustakuvaidea ei siis päätynyt ohjelman lopulliseen versioon.

Logon lisääminen päätettiin siis tehdä asettamalla erillinen kuva valmiin kuvaajan päälle. Tähän tarkoitukseen käytetään kuvankäsittelyyn ja graafisiin elementteihin erikoistuvan magick-paketin ominaisuuksia. Alkuperäisenä ajatuksena oli, että logo olisi osa kuvaajaa, jotta se piirtyisi myös pdf:ään. Pdf:ään ei kuitenkaan voi lisätä png-elementtiä, sillä vektorin ja bitti-

kartan yhdistäminen ei ole mahdollista käytössä olleilla työvälineillä. Magick ei myöskään kyennyt käsittelemään valmiita pdf-tiedostoja, joten kahden pdf:n yhdistäminen ei olisi ollut mahdollista. Näin ollen logon lisäämisestä pdf:ään luovuttiin ohjelmassa kokonaan.

Magickin image_composite()-komento mahdollisti kahden bittikartan yhdistämisen, ja sen pohjalta rakennettiin aiemmin luvussa kuvattu, lopullinen ratkaisu.

## 7.2 Selitteet

Selitetekstit eli labelit ovat tärkeä osa kuvaajia, joten on tärkeää, että valmis ohjelma näyttää ja sijoittaa ne oikein. Tässä luvussa käsittelen selitteisiin liittyvää ratkaisua, jolla niiden määrittely onnistuu pääohjelmasta käsin alaohjelmien määrää säästäen.

Pääohjelmassa selitteet määritellään muuttujien avulla. Selitteitä tulee olla oikea määrä piirrettävään dataan verrattuna, eli jos piirretään viivakuvaaja esimerkiksi Suomen ja Ruotsin lainakannoista, tulee selitteitä määritellä kaksi. Näiden selitteiden muuttujista kootaan lista “labels”, esimerkiksi näin: ”labels<- c(label1,label2)”. Tähän pääohjelmassa luotuun listaan viitataan alaohjelmissa, jolloin ggplot saa käyttöönsä aiemmin määritellyt selitteet. Esim: “ggplot(data=a_date, aes(x=Date, y=Value, colour= factor (SeriesName, labels = labels)), size = 3)”.

Ggplot osaa erotella yhden muuttujan perusteella sille annetut tiedot, vaikka ne olisivat samassa data framessa. Ggplotin peruskomennossa ggplot() sille annetaan käytettävä dataframe, sekä mitä muuttujia sen tulee käyttää akseleillaan ja minkä muuttujan perusteella sen tulee erotella dataa. Esimerkiksi: “ggplot(data=a_date, aes(x=Date, y=value, colour=SeriesName)”. Tällä komennolla ohjelma piirtäisi muuten onnistuneen kuvaajan, mutta selitteet tulisivat automaattisesti data framesta. Tässä tapauksessa selitteet siis olisivat aikasarjakoodeja, jotka eivät selkeyttäisi kuvaajan tulkitsemista lainkaan, vaan päinvastoin tekevät sen mahdottomaksi. Selitteet olisi siis hyvä määritellä erikseen, ja mieluiten pääohjelmassa muuttujia käyttäen, jotta alaohjelmia voisi kierrättää.

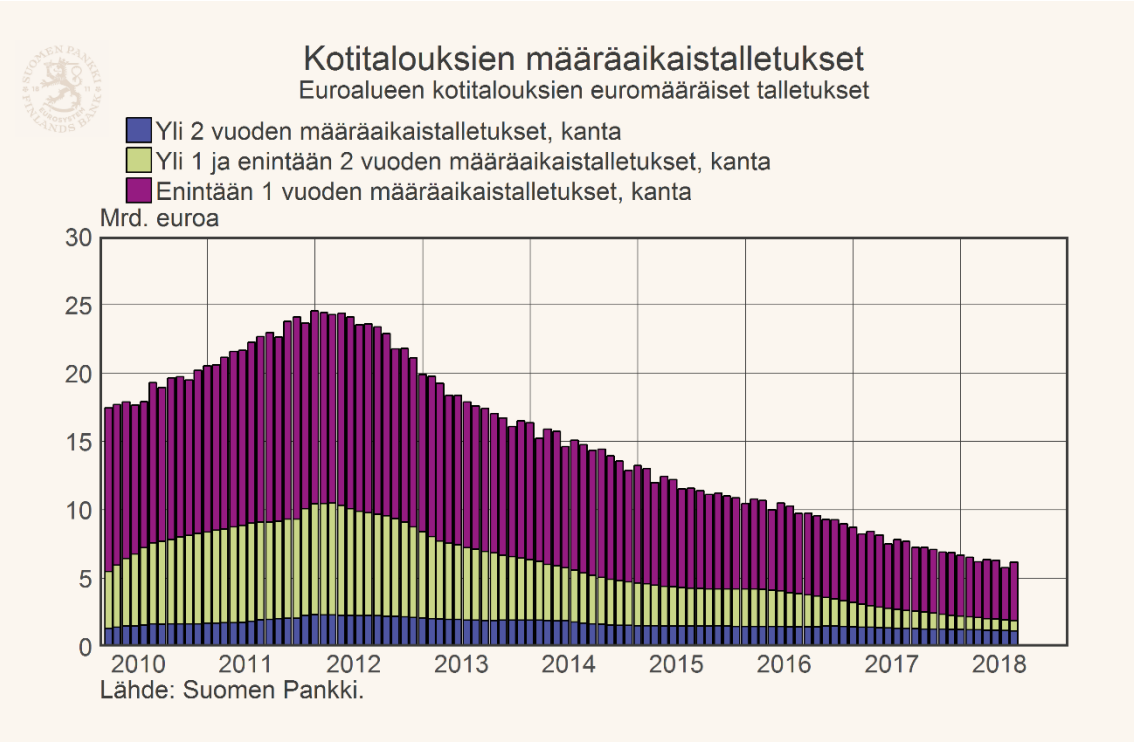
Selitteiden määrittäminen pääohjelmasta niin, että alaohjelmia pystyy tehokkaasti kierrättämään, oli yksi ensimmäisiä kehitystehtäviäni ohjelman parissa. Ohjelman rungossa oli jo luonnosteltu ratkaisua: selitteet määriteltäisiin muuttujilla, joihin viitattaisiin alaohjelmassa. Esimerkiksi pääohjelmassa määriteltäisiin selitteet “label1=”Suomi”” ja “label2=”Ruotsi””, ja alaohjelmassa niihin viitattaisiin näin: ”ggplot(data=a_date, aes(x=Date, y=Value, colour= factor (SeriesName, labels = list(label1,label2)), size = 3)) ”. Valmiissa kuvaajassa selitteet näkyisivät juuri sellaisina, kuin ne on pääohjelmassa kuvaajakohtaisesti määritelty.

Tämän lähestymistavan ongelma kuitenkin on, että ala- ja pääohjelmassa määriteltyjen selitteiden määrä on oltava sama. Esimerkin ratkaisua ei siis voisi käyttää kuvaajaan, joka tarvitsee kolme selitettä. Selitteitä ei samalla logiikalla voi myöskään lisätä alaohjelman listaan

mielivaltaisesti, ja olettaa, että ggplot käyttää niitä vain ikään kuin tarvitsemansa määrän. Alaohjelmia tarvitsisi siis tehdä useita perustuen siihen, kuinka monta selitettä kuvaajat tarvitsevat. Tätä kuitenkin nimenomaan haluttiin välttää, joten oli löydettävä uusi ratkaisu.

Koska seliteongelman ratkaisu oli yksi ensimmäisiä kehitystehtäviäni, enkä vielä täysin hallinnut R:n kaikkia ominaisuuksia, oli ensimmäinen ratkaisuehdotus hieman turhan monimutkainen. Idea oli kuitenkin sama kuin valmiin ohjelman ratkaisussa: pääohjelmassa koottaisiin tarvittavat selitteet yhteen, ja alaohjelmassa viitattaisiin niihin. Ensimmäisessä ratkaisussa selitteet listattiin “labels” listaan aivan kuin lopullisessa ratkaisussa, ja sen jälkeen sisällytettiin “lamount”-data frameen. Alaohjelman ggplotissa viitattiin data frameen näin: “labels = lamount$labels”. Myöhemmin ohjelmaa tarkastellessani kuitenkin huomasin, ettei selitteitä tarvinnut laittaa data frameen, vaan pelkän listan käyttö riitti piirtämään kuvaajiin oikeat selitteet. Muokkasin siis ratkaisun nykyiseen muotoonsa.

## 7.3 Datanpiirtojärjestys




Kuvio 7 Päällekkäisten pylväiden kuvaaja kotitalouksien määräaikaistalletuksista

Suurimmassa osassa Slaidisetin kuvia ei ole väliä sillä, missä järjestyksessä ggplot piirtää muuttujan erottelemat datan osat. Tähän on kuitenkin ohjelmassa yksi poikkeus: päällekkäiset pylväskuvaajat. Päällekkäisiä pylväskuvaajia piirrettäessä on selkeyden vuoksi tärkeää, että vakaina pysyvät tai hitaasti muuttuvat tiedot ovat kuvaajan pohjalla, ja vaihtelevammat tiedot taas päällä. Jos suuresti muuttuvat tiedot on piirretty kuvan pohjalle, on niiden päälle piirrettyjen tietojen tulkitseminen vaikeaa. Ggplotin automaattisesti piirtämä järjestys ei

aina ole optimaalisin, joten pääohjelmaan tuli kehittää ratkaisu, jolla pylväiden piirtämisjärjestystä voitaisiin hallita.

Selitteiden säätämisen yhteydessä oltiin jo huomattu, että ggplot piirtää eri tietojen kuvaajat aakkosjärjestyksessä sen muuttujan mukaan, jolla tiedot on eroteltu. Ratkaisun tulisi siis tehdä mahdolliseksi muuttaa tätä aikasarjakoodeihin perustuvaa järjestystä.

Ratkaisuna luodaan total-data frameen uusi muuttuja “variable”, jonka mukaan data erotellaan ja piirretään kuvaajaan. “Variablen” arvot ovat aakkosia, jotka määritellään pääohjelmassa. Jokaista uniikkia aikasarjakoodia vastaa yksi aakkonen, joten periaatteessa erottelu tapahtuu yhä aikasarjakoodiin perustuen. Antamalla tietylle aikasarjakoodille tietyn aakkosen voi määrittää sitä, missä järjestyksessä siihen liittyvät tiedot piirtyvät kuvaajaan. Pääohjelman ratkaisu toimii siis näin: “index <- c("Aikasarja1", "Aikasarja2", "Aikasarja3", "Aikasarja4", "Aikasarja5", "Aikasarja6", "Aikasarja7")

values<- c("B","A","C","D","E","F","G")

total$variable <- values[match (total$SeriesName, index)]”.

Esimerkissä “Aikasarja2”:n tiedot piirtyisivät kuvaajan pohjalle, “Aikasarja1”:n tiedot niiden päälle, “Aikasarja 3”:n tiedot taas niiden päälle, ja niin edespäin. Uudelleenjärjestämistä tarvitseville kuvaajille on oma alaohjelmansa, geom_bar_stack_order.R, jossa SeriesNamen sijaan käytetään variablea tiedon erotteluun.

Tähän ratkaisuun ei varsinaisesti liity merkittäviä kehitysvaiheita, vaan yllä kuvailtu ratkaisu oli ensimmäinen ja ainut, jota testattiin. Koska se kuitenkin on melko monimutkainen soveltaa uutta kuvaajaa koodatessa ja vaatii harkintaa ja tarkkuutta, olen kirjoittanut tarkat ohjeet ratkaisun käyttämisestä Slaidisetin käytön ohjeistukseen.

## 7.4 Toinen Y-akseli

Toisen y-akselin lisääminen kuvaajaan on ollut kehitystehtävistä pitkäkestoisin ja vaiheikkain. Ggplotilla toisen akselin lisääminen ei ole mahdotonta, mutta melko haastavaa. Tämä on täysin tarkoituksellista, sillä kahden eri akselin käyttämistä samassa kuvaajassa ei yleisesti suositella. Toista y-akselia kuitenkin nimenomaan toivottiin eräisiin Slaidisetin kuvaajiin, joten ratkaisua lähdettiin kehittämään.

Toisen y-akselin piirtäminen vaatii muokkauksia niin pää- kuin alaohjelmaankin. Pääohjelmassa määritellään toisen y-akselin parametrien muuttujat, ja alaohjelmassa se piirretään lisäämällä ggplot-komentoon scale_y_continuous()-komennon sisään sec_axis()-komento. Myös data framen tietoja tulee muokata, jotta ne piirtyisivät oikein suhteessa oikeanpuoleiseen y-akseliin. Kahden y-akselin kuvassa on tärkeää, että molempien akseleiden asteikot käyttävät

yhtä montaa vaakaviivaa. Asteikot siis voivat olla erilaisia, esimerkiksi 0-5 vasemmalla ja 0-10 oikealla, mutta käytettävien vaakaviivojen määrä tulee olla sama, jotta asteikkojen numerot olisivat niiden kohdalla. Esimerkissä siis jommankumman asteikon välejä tulisi muuttaa, esimerkiksi vasemmanpuoleisella asteikolla luvut olisivat puolikkaina, tai oikeanpuoleisella asteikolla vain joka toinen luku näkyisi.

Alaohjelmaan lisätty sec_axis()-komento lisää kuvaajan oikealle laidalle toisen y-akselin. Komento näyttää kokonaisuudessaan tältä: “sec.axis = sec_axis(trans = ~./y2_scale, breaks = break2)”. Trans-parametrilla muokataan vasemmanpuoleisen y-akselin ominaisuuksista uusi y-akseli oikealle, ja breaks-parametrilla määritetään sen asteikko. Muuttujat y2_scale ja break2 määritellään pääohjelmassa, kuten muutkin kuvaajien yksilölliset ominaisuudet.

Pääohjelmassa määriteltävät parametrien muuttujat ovat y2_max, y2_scale ja break2. Ensimmäiseksi mainittua ei varsinaisesti käytetä alaohjelmassa asteikon määrittelyyn, vaan tietojen muuttamiseen niin, että ne asettuvat kuvaajassa oikein suhteessa oikeanpuoleiseen akseliin. Oikeanpuoleinen y-akseli ei ole ggplotissa varsinainen akseli samalla tavalla kuin vasemmanpuoleinen, eikä sitä voi hallita samalla tavalla. Kaikki kuvaajan tiedot piirtyvät suhteessa vasemmanpuoleiseen akseliin, joten jos akselit poikkeavat toisistaan, ei jää muuta vaihtoehtoa kuin muokata niitä tietoja, joiden tulisi olla sidoksissa oikeaan y-akseliin. Ohjelmassa tämä on toteutettu kertomalla tiedot y-akseleiden maksimien osamäärällä: “df1$Value<-df1$Value*(y_max/y2_max)”.

Y2_scale taas on luku, jota käytetään trans-parametrissa alaohjelmassa. Sillä ikään kuin jaetaan vasen y-akseli, kuten komennossa näkyy. Jos esimerkiksi vasemman y-akselin maksimi on 18, ja oikeanpuoleiselle akselille halutaan maksimi 9, y2_scalen arvoksi tulee laittaa 2. Jos taas tilanne olisi päinvastoin ja maksimi haluttaisiinkin kertoa, käytetään käänteislukua eli arvoksi tulisi 0,5. Break2 sisältää lukujonon, jossa listataan akselin asteikon minimi, maksimi ja väli.

Kaikkia kahden y-akselin kuvaajia tämä ei koske, mutta esimerkiksi viiva- ja palkkikuvaajien yhdistelmässä käytetään eri data frameja erityyppisten kuvaajien piirtämiseen. Alaohjelmassa ggplotissa viitataan ensin palkkien tiedot sisältävään data frameen, ja sen jälkeen geom_line()-komennossa viivan tiedot sisältävään erilliseen data frameen.

Manuaalisesti koottu Slaidisetti, joka toimii automatisoidun Slaidisetin mallina, sisältää kahden y-akselin kuvaajia, joten luonnollisesti niitä pyrittiin tekemään myös automatisoidussa ratkaisussa. Vaikka tällainen ominaisuus saatiinkin sisällytettyä valmiiseen ohjelmaan, projektin aikana oli vaiheita, joissa kahden y-akselin kuvaajien tarpeellisuus kyseenalaistettiin, ja niille pohdittiin vaihtoehtoisia toteutustapoja.

Kun aloin kehittämään ratkaisua, selvisi pian, että toisen y-akselin lisääminen kuvaajaan ei ole ensinnäkään yksinkertaista, eikä yleisesti suositeltavaakaan. Tässä vaiheessa kokeiltiin ratkaisua, jossa kahden y-akselin eri kuvaajat erotettiin omiksi kuvaajikseen. Verkkojulkaisukokouksessa kuitenkin päädyttiin siihen lopputulokseen, että erotellut kuvat eivät olleet miellyttävän näköisiä eivätkä palvelleet tarkoitustaan optimaalisesti.

Ennen kahden y-akselin ratkaisun kehittämistä kokeiltiin vielä kahden kuvaajan asettamista samalle sivulle, jotta niiden tietoja olisi mahdollista vertailla. Tästä kuitenkin luovuttiin, sillä kuvaajat eivät näyttäneet miellyttäviltä ja vertailukelpoisilta päällekkäin eikä vierekkäin. Tultiin siihen lopputulokseen, että kahden y-akselin kuvaaja on paras ratkaisu kahden erityyppisen kuvaajan keskinäiseen vertailuun.

Vaikka sec.axis-komentoa oltiin kokeiltu jo projektin alkuvaiheessa, toisen akselin asteikon ja sen mukaan asettuvien tietojen muokkaamista ei oltu kehitetty kovinkaan pitkälle. Näiden ominaisuuksien haastava toteuttaminen olikin yksi niistä syistä, miksi kahden y-akselin kuvaajista suunniteltiin luovuttavan ohjelmassa. Tutustuttuani kuitenkin paremmin trans-parametrin toimintaan, lisäämällä breaks-parametrin ja testaamalla molempia sain kehitettyä aiemmin kuvaillun ratkaisun toisen y-akselin asteikon määrittämisestä. Asteikon määrittäminen ei ole ohjelman käyttäjälle edelleenkään niin yksinkertaista kuin vasemman y-akselin tapauksessa, mutta se onnistuu silti ilman useita alaohjelmia sekä niin, että lopputulos on esteettisesti miellyttävä ja tarkoituksenmukainen.

## 7.5 Tallentaminen

Kuvien tallentaminen tulee luonnollisesti olla yksi ohjelman ominaisuuksista. Grafiikkaa halutaan tarkastella muutenkin kuin vain hetkellisesti R:n käytön yhteydessä, joten ohjelman on kyettävä tallentamaan kuvaajat. Vaikka RStudio tarjoaa mahdollisuuden tallentaa sillä piirrettyjä kuvaajia kuvatiedostoina, on koodattu ratkaisu silti automaation kannalta parempi vaihtoehto.

Ohjelma sisältää usean eri tallennusratkaisun, ja sitä ajettaessa kuvat tallentuvat useaan kertaan ja eri muodoissa. Logon lisäämistä käsittelevässä luvussa on kuvailtu, miten kuvaajien tallentaminen png-kuviksi on ratkaistu: kuvaaja tallennetaan ggsave()-komennolla, jonka jälkeen tallennettu kuva yhdistetään logoon. Logon ja kuvaajan yhdistelmä tallennetaan alkuperäisen, logottoman kuvan päälle.

Pdf-tallennus toimii luomalla lista kuvaajille, lisäämällä kuvaajat siihen alaohjelmissa, ja lopuksi kokoamalla ne yhteen ja tallentamalla ne pdf:nä. Ennen kuvaaja-osiota ohjelma luo listan “graphs” komennolla “graphs = list()”. Alaohjelmat sisältävät komennon “graphs[[length(graphs)+1]] = g”, joka lisää piirretyn kuvaajan graphs-listaan. Kun kaikki kuvaajat on piirretty, pääohjelman lopussa on komento “multiplot = marrangeGrob(graphs,

nrow=1, ncol=1, top=NULL)”, joka luo objektin “multiplot”. MarrangeGrob()-komento järjestää graphsin kuvaajat yhdeksi kokonaisuudeksi. Tämän jälkeen multiplot tallennetaan pdf:nä ggsave()-komennolla.

Alkuperäinen Slaidisetti on PowerPoint-esitys, joten myös sen mahdollisuutta harkittiin projektin alussa. Toteutusvaihtoehtoja tutkittaessa kuitenkin kävi ilmi, että kuvaajien vieminen PowerPointiin olisi haastavaa ja vaatisi usean uuden paketin asentamista. Pdf oli PowerPoint-esitykseen verrattuna helpompi toteuttaa, ja myös muuten toivotumpi ratkaisu.

R:n erilaiset paketit mahdollistavat sen, että samaa tarkoitusta varten voi olla olemassa useampi eri toiminto. Kun pdf:n tallentaminen ensi kertaa sisällytettiin ohjelmaan, se tehtiin eri komennolla kuin lopullisessa versiossa. Tämä komento myös toimi hieman eri tavalla, kuin graphs-listan kokoamiseen perustuva ratkaisu. Komento “pdf(file=””)” sijoitettiin ohjelman alkuun, ja sen pysäyttävä “graphics.off”-komento taas aivan ohjelman loppuun. Komento avaa graphics devicen, joka tulostaa kuvaajista pdf-tiedoston siihen saakka, kun se suljetaan. Tämä ratkaisu ei kuitenkaan tukenut ulkoista grafiikkaa eli textGrobia, joka on tärkeä lisäys kuvaajaan. Niinpä ohjelmassa päädyttiin käyttämään aiemmin kuvattua, joustavampaa ratkaisua.

Projektin aikana on testattu myös kuvien automaattista tallentamista Suomen Pankin intraan. Tätä ominaisuutta ei kuitenkaan lopullisessa ohjelmassa ole, sillä R:n kautta tallennetut tiedostot eivät auenneet intrassa. Ratkaisun ideana oli, että kuvien tallennuskoodiin yksinkertaisesti sisällytettäisiin polku tallennuskansioon intrassa. Tiedostot ilmestyivätkin intraan, mutta ne näki vain ne lisännyt henkilö, eikä niitä voinut avata. Ratkaisu ei kuitenkaan ollut aivan turha, sillä sen pohjalta kehitettiin lopullisen ohjelman tallennusratkaisu, joka hyödyntää muuttujien avulla muokattavia polkuja ja kuvanimiä.

# 8 Jatkokehittäminen

## 8.1 Jatkokehittämisen tarve

Työskentelin projektin parissa rajallisen ajan, jolloin pyrin työstämään ohjelmointiratkaisua niin pitkälle kuin mahdollista. Projektin edetessä kuitenkin tuli selväksi, ettei ohjelma tulisi täysin käyttövalmiiksi elokuun alkuun mennessä, jolloin työsuhteeni päättyisi. Suurin syy tähän oli aikasarjakoodien puute: kaikille Slaidisetissä perinteisesti esitetyille tilastoille ei oltu vielä perustettu aikasarjoja, joista getdata-funktio hakisi tiedot ohjelman piirrettäväksi. Slaidisetissä ei siis olisi vielä tarpeeksi kuvaajia, vaan niitä tulisi lisätä ohjelmaan sitä mukaa, kun aikasarjoja perustettaisiin. Tästä johtuen päätin ottaa ohjelman jatkokehityksen mahdollistamisen ja helpottamisen yhdeksi tavoitteekseni projektissa. Tässä luvussa esitellään niitä keinoja, joilla olen pyrkinyt helpottamaan ohjelman käyttöä ja jatkokehitystä, jotta automaatioprojekti saataisiin päätökseensä.

## 8.2 Ohjeistus

Jatkokehityksen suurin prioriteetti on uusien kuvaajien lisääminen Slaidisettiin. Ajatuksena oli, että ohjelman koodiin lisättäisiin uusia kuvaajia sitä mukaa, kun tarvittavia aikasarjoja perustettaisiin. Uusien kuvaajien lisääminen ohjelmaan voi kuitenkin osoittautua monimutkaiseksi käyttäjälle, joka ei ohjelmaan aikaisemmin tutustunut tai ollut sen kehittämisessä mukana. Niinpä päätin kirjoittaa ohjeet, jotka neuvovat niin Slaidisetti-ohjelman normaalin ajamisen vaiheet, kuin myös sen, miten lisätään uusi kuvaaja.

Ohjeet on toteutettu Suomen Pankin Word-ohjepohjalle, ja ne sijaitsevat ohjelman kanssa samassa kansiossa. Ohjeessa käydään ensin yksityiskohtaisesti ja vaihe kerrallaan läpi ohjelman normaaliin ajamiseen liittyvät vaiheet. Vaikka ohjelman koodi sisältää runsaasti kommentteja ohjelman ajamiseen, on uudelle käyttäjälle hyvä olla olemassa myös hieman yksityiskohtaisemmat ohjeet, jotka myös selkeyttävät ohjelman toimintaa kokonaisuutena. Ohjeiden avulla myös R:ää taitamattoman käyttäjän tulisi olla mahdollista ajaa ohjelma onnistuneesti.

Uuden kuvaajan lisäämistä käsittelevässä osiossa on myös pyritty mahdollisimman yksityiskohtaiseen ja perinpohjaiseen ohjeistukseen, sillä vaihe vaatii käyttäjältä harkintaa ja tarkkuutta. Kaikki kuvaajien muuttujat on listattu ja niiden toiminta on selostettu. Monimutkaisemmat prosessit, kuten toisen y-akselin asettaminen ja piirtojärjestyksen muuttaminen, on kuvattu käyttäen apuna kuvia. Ohje listaa myös alaohjelmat, ja selostaa niiden tarkoitukset sekä niissä käytettävät muuttujat. Alaohjelmien lista sisältää myös kuvalliset esimerkit niiden piirtämistä kuvista.

Ohjeet on esitelty ohjelman esittelytilaisuudessa, ja käyty myös läpi yhdessä projektista vastaavan Ville Tolkin kanssa. Ohjeita luonnollisesti päivitetään, jos ohjelman perusrakenteeseen tai muihin ohjeissa määriteltyihin ohjelman osiin tehdään muutoksia. Muutokset merkitään selkeästi päivämäärällä.

# 9 Palaute

Automaatioratkaisu esiteltiin Suomen Pankin tilastoyksikölle 9.8.2018. Tilaisuudessa testattiin ohjelman toimintaa käytännössä, ja esiteltiin käyttöohjeet. Tilaisuudessa vastattiin myös yleisön kysymyksiin, ja keskusteltiin ohjelman ominaisuuksista. Tilaisuuden aikana saatu palaute oli positiivista. Yksi esille noussut kehitysehdotus oli, että ohjelmaa kehitettäisiin excel-painotteisemmaksi. Ehdotettiin, että kuvaajille yksilöllisiä parametrien arvoja, kuten otsikoita ja akseleita, voisikin säädellä syöttämällä arvot ulkoiseen excel-tiedostoon. Tällä ratkaisulla voitaisiin yksinkertaistaa esimerkiksi uuden kuvaajan lisäämisprosessia käyttäjän näkökulmasta. Toistaiseksi tällaista ominaisuutta ei olla kuitenkaan vielä suunniteltu lisättäväksi ohjelmaan.

Automaatioratkaisusta saatu palaute on ollut pääosin positiivista. Varsinainen kritiikki on tullut projektin aikana kehitysehdotusten muodossa, ja niiden pohjalta on jatkettu ohjelman kehittämistä. Kehitysehdotusten pohjalta tehtyjä ominaisuuksia ovat esimerkiksi kahden y-akselin kuvaajat, logon lisääminen kuviin, sekä visuaalisen ilmeen hiominen Suomen Pankin linjan mukaiseksi.

# 10 Arviointi

Projektin lopputuotoksen onnistuneisuutta voidaan mitata vertaamalla sitä projektin alussa asetettuihin tavoitteisiin. Kuten ohjelman kuvauksesta ja ominaisuuksien esittelystä on käynyt ilmi, automaatioratkaisu sisältää kaikki toivotut ominaisuudet datan hakemisesta valmiin tilastografiikan tallentamiseen. Raportissa on toisaalta myös kerrottu ohjelman ominaisuuksiin tehdyistä kompromisseista, ja siitä, ettei ohjelma ole varsinaisesti vielä käyttövalmis. Toisaalta taas kompromissit eivät kohdistuneet ohjelman tärkeimpiin ominaisuuksiin, vaan esimerkiksi logon lisäämiseen ja kuvaajien tallentamiseen suoraan intraan. Myös ohjelman keskeneräiseksi jäämiseen varauduttiin projektin aikana jatkokehitykseen panostamisella.

Lopputuotoksen onnistuneisuutta voidaan mitata myös arvioimalla, onko toimeksiantaja hyötynyt siitä. Tähän mennessä ratkaisua ei ole vielä keskeneräisyyden vuoksi otettu varsinaiseen käyttöön, mutta palaute on ollut positiivista. Ohjelmaa on esittelytilaisuuden jälkeen jatkokehitetty lisäämällä niin uusia ominaisuuksia kuin uusia kuvaajiakin.

Lähteet

Painetut

Gentleman, R. & Ihaka, R. 1996. R: A Language for Data Analysis and Graphics. Journal of Computational and Graphical Statistics 3/1996. Oxfordshire: Taylor & Francis Ltd., 299-314.

Gohil, A. 2015. R Data Visualization Cookbook. Birmingham: Packt

Hildén, J., Koponen, J. & Vapaasalo, T. 2017. Tieto näkyväksi. Helsinki: Aalto-yliopisto.

Holopainen, M. & Pulkkinen, P. 2012. Tilastolliset menetelmät. 5.-7. painos. Helsinki: Sanoma Pro.

Karjalainen, J. & Karjalainen, L. 2009. Tilastojen graafinen esittäminen. Keuruu: Pii-Kirjat.

Lanz, B. 2013. Machine Learning with R. Birmingham: Packt

Lim, A. & Tjhi, W. 2015. R High Performance Programming. Birmingham: Packt

Rahlf, T. 2014. Data Visualisation with R. Cham: Springer.

Sähköiset

CRAN.R-project. 2015. The Comprehensive R Archive Network. Viitattu 3.12.2018. https://cran.r-project.org/

CRAN.R-project. 2018. magick: Advanced Graphics and Image-Processing in R. Viitattu 3.12.2018. https://cran.r-project.org/web/packages/magick/index.html

Grolemund, G. & Wickham, H. 2017. R for Data Science. Viitattu 3.12.2018. https://r4ds.had.co.nz/

R-Project. 2018. About R. Viitattu 3.12.2018. https://www.r-project.org/about.html

R-Project. 2018. Contributors. Viitattu 3.12.2018. https://www.r-project.org/contributors.html

RDocumentation. 2018. Ggplot2. Viitattu 3.12.2018. https://www.rdocumentation.org/packages/ggplot2/versions/3.1.0

RDocumentation. 2018. Create Grid Graphical Objects, aka "Grob"s. Viitattu 3.12.2018. https://stat.ethz.ch/R-manual/R-devel/library/grid/html/grid.grob.html

Wickham, H. 2018. Introduction. Viitattu 3.12.2018. http://r-pkgs.had.co.nz/intro.html

Wickham, H. 2018. R Packages. Viitattu 3.12.2018. http://r-pkgs.had.co.nz/

Julkaisemattomat

Kuviot

Liitteet

Liite 1: Slaidisetti R-ohjelman käyttöohje

**Slaidisetti R-ohjelman käyttöohje**

**Johdanto**

Tämä on Slaidisetti R-ohjelman käyttöohje, joka kertoo ohjelman ominaisuuksista, sekä sen ajamisesta ja muokkaamisesta.

Ohjelma automatisoi tilastografiikkaa sisältävän Slaidisetin hakemalla tilastotiedot aikasarja-koodin avulla, ja piirtämällä niiden pohjalta tilastografiikkaa. Kuvat tallentuvat yksittäisinä png-tiedostoina, sekä yhtenä koottuna pdf-tiedostona.

**2 Valmistelu**

**2.1 Tiedostot**

Käynnistä R tai R-Studio.

Hae ohjelma/-t:

Pää- ja alaohjelmat, sekä kuvien kansiot, löytyvät polusta:

**X:\RMTI\RMTITA\Slaidisetti**

Tällä hetkellä käytössä oleva pääohjelma on:

**000_SlaidisettiMain_v2.R**

**2.2 Paketit**

Pääohjelman alussa "VALMISTELU"-osiossa on listattu ohjelman toimimiseen tarvittavat r-paketit. Aja kaikki library(Paketin nimi)-komennot ottaaksesi paketit käyttöön. Jos sinulla ei ole vielä paketteja asennettuina, voit tehdä sen install.packages("Paketin nimi")-komennolla, joka löytyy library()-komentojen jälkeen kommenttina. Seasonal-paketin ottaminen käyttöön saattaa vaatia lupaa järjestelmänvalvojalta.

**2.3 Työkansiot ja funktiot**

Aja setwd(" ")-komento "VALMISTELU"-osiossa asettaaksesi työkansion.

Seuraavaksi määritellään polku siihen kansioon, mihin Slaidisetin kuvat tallentuvat. Tämä on ainut osa koodia, jota sinun tarvitsee ohjelman normaalin ajamisen yhteydessä muokata. Vaihda kk ja vv-parametreihin kuukausi- ja vuosiluku MM ja YYYY- muodossa ja aja koodi. Koodi johtaa näistä pvm-parametrin ja hyödyntää vv-parametria end1-parametrin määrittelyssä kullekin kuvaajalle. Aja slaidipolku <- paste0("X:/RMTI/RMTITA/Slaidisetti/Slaidit/",pvm,"/"), ja aja sitten polun alapuolella oleva tarkistuskoodi. Koodi luo polun mukaisen kansion, jos sitä ei ole vielä olemassa, ja muussa tapauksessa antaa virheilmoituksen. Mikäli haluat muokata jo olemassa olevan kansion sisältöä, jätä tämä vaihe väliin.

Aja vielä "VALMISTELU"-osion source(" ")-komennot ottaaksesi bof_themen ja getdata-funktion käyttöön.

**3 Ohjelman ajaminen**

Tämän jälkeen voit ajaa kaikki kuvaajat kerralla, tai ajaa vain yksittäisten kuvien koodeja. Yksittäiset kuvien koodit alkavat ristikkomerkein ympäröidyistä numeroista ja päättyvät p-

funktioon. Huomaa kuitenkin, että pdf:ää varten on ajettava koko koodi graphs = list()-komennosta graphics.off()-komentoon asti. Kuvat tallentuvat pdf:ään siinä järjestyksessä, missä niiden koodit ovat ajettu pääohjelmasta.

## 4 Kuvien lisääminen ja muokkaaminen

### 4.1 Uuden kuvan lisääminen

Jos sinun tarvitsee lisätä Slaidisettiin uusi kuva, on hyvä käyttää pohjana valmista koodia. Jos esimerkiksi tarvitaan uusi viivakuvaaja, voit kopioida "Yleisön lainakannan kasvu"- kuvan koodin ja liittää sen siihen kohtaan koodia, johon haluat sen tallentuvan pdf:ssä. Tämän jälkeen tarvitsee muuttaa kopioidun koodin parametreja.

Mikäli kuvien numerointia tarvitsee muuttaa, muistathan selkeyden vuoksi muuttaa niin ristikkomerkein ympäröityä numerointia, kuin myös kuvan nimeä (kuvio="").

### 4.2 Parametrit: Ruudukko ja otsikot

Piirrettävän kuvan visuaalisia ominaisuuksia voi muuttaa muokkaamalla kuvakoodien alkuun sijoittuvia parametreja. Alla lista näistä parametreista ja niiden ominaisuuksista.

**start1 ja end1**

Aikaväli, jolta tiedot kuvaajan haetaan ja piirretään. Tulee olla muodossa "YYYY-MM-DD". Loppupäivämäärä generoidaan automaattisesti hyödyntäen "VALMISTELU"-osiossa määriteltyä vv-parametria. Käytetään kaikissa kuvaajissa kuluvan vuoden viimeistä päivää. Alkupäivämäärää voi vaihtaa tarvittaessa.

**y_max, y_min ja by**

Vasemmanpuoleisen Y-akselin maksimi, minimi, ja välit. Oikeanpuoleisen Y-akselin minimien, maksimien ja välien määrittelystä kerrotaan omassa luvussaan.

**xaxis**

Määrittää sen, näytetäänkö X-akselilla vuosia (%Y) vai kuukausia (%b).

**linesize**

Viivan paksuus. Yleensä 2, mutta paljon tietoa sisältävää kuvaajaa tehtäessä voi selkeyden vuoksi olla hyvä ratkaisu ohentaa viivaa hieman.

**kuvio, title, ja subtitle**

Png-kuvan nimi, kuvion otsikko sekä alaotsikko.

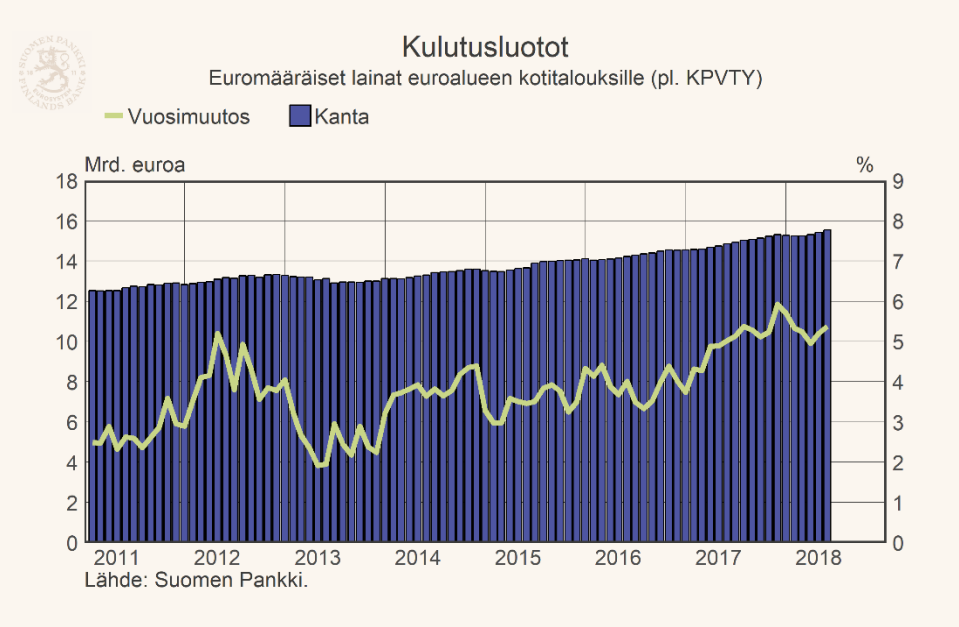
**xtitle ja ytitle (y2title ja titles)**

Akseleiden otsikot. X-akselilla ei yleensä ole otsikkoa. Y2title=""-parametrilla muokataan kahden y-akselin kuvaajassa oikeanpuoleisen y-akselin otsikkoa. Titles=""-parametri yhdistää y-akseleiden otsikot, eikä sitä tarvitse muokata.

**caption**

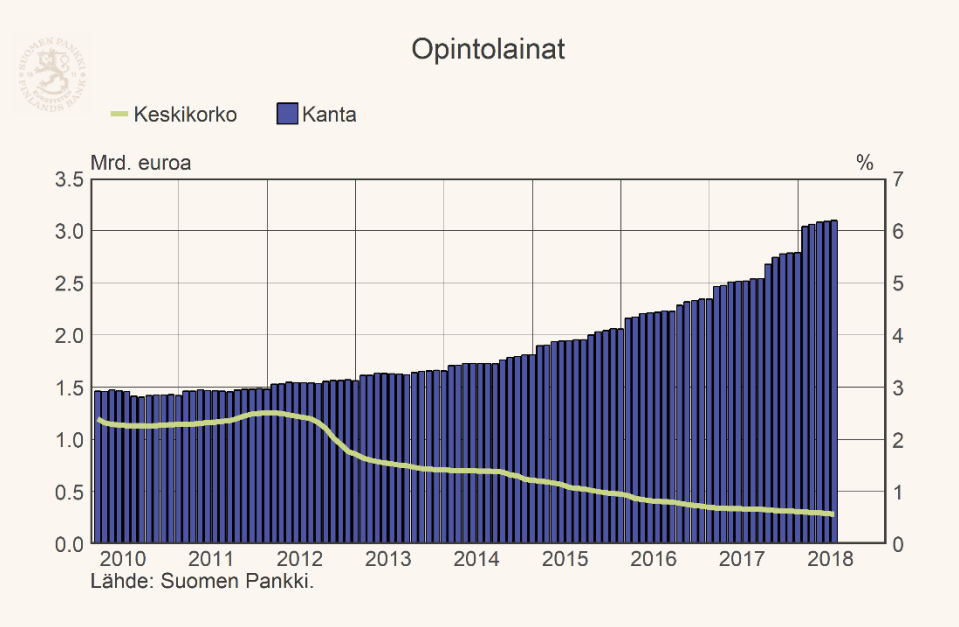
Kuvan alakulman teksti, jossa kerrotaan lähteet.

### 4.3 Ruudukko kahden y-akselin kuvassa




```
#Määritellään toisen y-akselin asteikkoa
y2_max=9
y2_scale=2
break2<- seq(0,9,1)
```

```
#Määritellään toisen y-akselin asteikkoa
y2_max=7
y2_scale=0.5
break2<- seq(0,7,1)
```




Kahden y-akselin kuvassa parametreja on enemmän, ja akselien asettumista kohdilleen on hieman suunniteltava. Tavoitteena on, että numerot ja asteikko asettuvat kohdilleen kuten esimerkkikuvissa. Tämä onnistuu silloin, kun oikean y-akselin maksimiluku on jaettavissa tai kerrottavissa vasemman y-akselin maksimiluvuksi. Oikeanpuoleisen y-akselin arvojen määrittely ja asettelu tapahtuu muokkaamalla "#Määritellään toisen y-akselin asteikkoa"-kommentin alla olevilla parametreja:

**y2_max=**

Oikeanpuoleisen y-akselin asteikon maksimiluku. Ei varsinaisesti määritä y-akselin asteikkoa, vaan käytetään kertomaan oikeanpuoleista asteikkoa käyttävät tiedot niin, että ne asettuvat kuvassa akseliin nähden oikein. Tämä siksi, että kaikki kuvissa käytettävä tieto asettuu automaattisesti vasemman y-akselin mukaan.

**y2_scale=**

Luku, jolla vasemman y-akselin maksimi jaetaan, esim. Kulutusluotot-kuvassa 18/**2**=9. Jos luku pitää kertoa, kuten esim. Opintolainat-kuvassa, käytetään vastalukua.

**break2<- seq( , , )**

Oikeanpuoleisen y-akselin minimi, maksimi, sekä välit.

### 4.4 Parametrit: Labelit

Labeleita ja niiden määrää muokataan "#labels in legend" kommentin alapuolella. Jos esimerkiksi tarvitset kolme labelia, mutta pohjana käyttämässäsi koodissa on vain kaksi, lisää koodiin uusi rivi "**label3 = "Label tähän"**". Voit myös muokata sitä, monellako rivillä labelit näkyvät **lrow=**-parametrillä. Kun lisäät tai poistat labeleita, muista myös muokata alempaa löytyvää "labels<- c(label1,label2)"-koodia niin, että se vastaa antamiasi labeleita, esim. labels<- c(label1,label2,label3).

Jos teet kahden y-akselin kuvaajan, muista mainita labelissa, kumpaa asteikkoa (oikea vai vasen), muuttuja käyttää.

Labelit on asetettava aakkosjärjestykseen sen muuttujan perusteella, jota käytetään kuvassa tietojen erottelemiseen. Useimmiten tämä muuttuja on SeriesName, eli se aikasarjakoodi, jolla tiedot on haettu. Idea tulee luultavasti helpommin esille seuraavan sivun esimerkkikuvasta.

```
# labels in legend - tulee olla aakkosjärjestyksessä SeriesNamen mukaan, mu
label1 = "Saksa"
label2 = "Espanja"
label3 = "Suomi"
label4 = "Ranska"
label5 = "Italia"
label6 = "Euroalue"
lrow=1

#Luodaan 'lamount' dataframe ja laitetaan sinne tarvittavat labelit
labels<- c(label1,label2,label3,label4,label5,label6)
```

```
#poimitaan data
df1<-data.frame(getdata(          .FI.                    )) #Suomi
df2<-data.frame(getdata(          .U2.                    )) #Euroalue
df3<-data.frame(getdata(          .DE.                    )) #Saksa
df4<-data.frame(getdata(          .FR.                    )) #Ranska
df5<-data.frame(getdata(          .IT.                    )) #Italia
df6<-data.frame(getdata(          .ES.                    )) #Espanja
```

Kuvaajien piirtojärjestyksen muokkaamisesta tarvittaessa kerrotaan käyttöohjeessa myöhemmin.

### 4.5 Tietojen hakeminen ja muokkaaminen

Kuvissa käytettävät tiedot haetaan aikasarjahaulla getdata-funktiota käyttäen. Aikasarjat haetaan erikseen ja sisällytetään omiin data frameihinsa näin:

**df1<-data.frame(getdata("Aikasarjakoodi tähän"))**

**df2<-data.frame(getdata("Toinen koodi tähän"))**

Useimmissa kuvissa nämä data framet yhdistetään yhdeksi data frameksi:

**total<-rbind(df1,df2)**

Esimerkiksi kuvissa, joissa on sekä viiva- että palkkikuvaaja, data framet pidetään kuitenkin erillään.

Jotta data framen tiedot piirtyisivät kuviin toivotulla tavalla, tulee niitä hieman muokata. Tiedot pitkään muotoon muuttavaa koodia ei tarvitse muokata, mutta joskus tulee tarpeeseen muokata data framen tietoja, jotta ne asettuisivat kuvaan oikein. Esimerkiksi miljardeja euroja käsittelevissä kuvissa tiedot on jaettava tuhannella:

**total$Value<-total$Value/1000**

0_SlaidisettiMain_v2.R × | total ×

Filter

| | DatasetID | SeriesName | Period | Value |
|---|---|---|---|---|
| 1 | | | 1997-09-30 | 25543.77 |
| 2 | | | 1997-10-31 | 25696.74 |
| 3 | | | 1997-11-30 | 26063.58 |

000_SlaidisettiMain_v2.R × | a_long ×

Filter

| | Date | variable | Value |
|---|---|---|---|
| 1 | 1997-09-30 | B | 25.54377 |
| 2 | 1997-10-31 | B | 25.69674 |
| 3 | 1997-11-30 | B | 26.06358 |

### 4.6 Piirtojärjestyksen muuttaminen

Kuten aiemmin on mainittu, ggplot piirtää kuvaajat aakkosjärjestyksessä muuttujan, useimmiten SeriesNamen, mukaan. Joissain kuvissa, kuten esimerkiksi päällekkäisten palkkien kuvaajissa, tämä järjestys ei kuitenkaan aina ole paras mahdollinen. Jos järjestystä haluaa muuttaa, on luotava aikasarjakoodeihin perustuva uusi muuttuja (esimerkissä "variable"), jolla tiedot voidaan asettaa uuteen aakkosjärjestykseen.

**index <- c("Aikasarjakoodi1", "Aikasarjakoodi2", "Aikasarjakoodi3")**

**values<- c("C","B","A")**

**total$variable <- values[match (total$SeriesName, index)]**

Tiedot piirtyvät nyt järjestyksessä Aikasarja3->Aikasarja2->Aikasarja1

Huomaathan, että koska muuttuja ei tällä kertaa ole SeriesName, on kuvan piirtämiseen käytettävä toista alaohjelmaa, esim. **geom_bar_stack_order.R.**

### 4.7 Alaohjelma

Kuvien piirtämiseen käytetään erilaisia alaohjelmia sen mukaan, millainen kuva halutaan piirtää. Käytettävä alaohjelma näkyy kuvan koodin loppuosassa, esim:

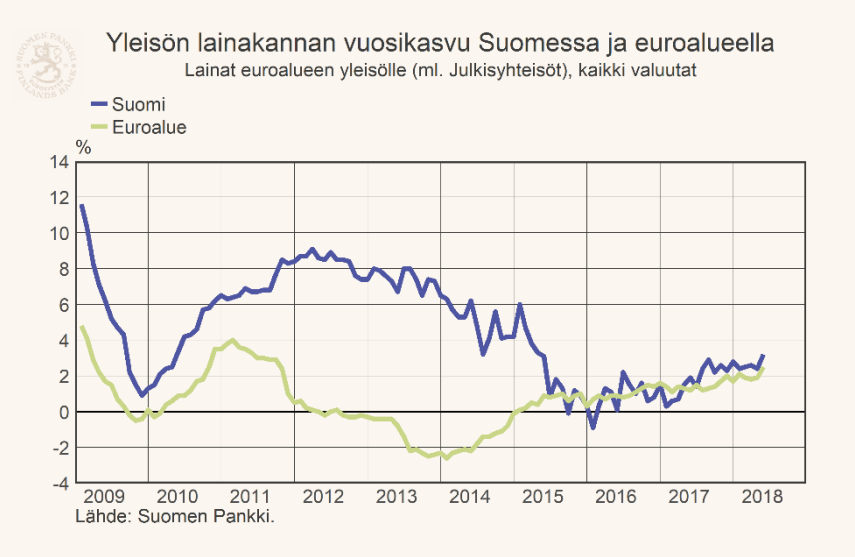
**source("Geom_line_longv4.R")**

Excel-tiedostossa "Lista geom-tiedostoista" on lueteltu alaohjelman ja niiden ominaisuudet.

Jos sinun täytyy muokata olemassa olevaa alaohjelmaa, luo mieluiten sen pohjalta kokonaan uusi alaohjelma, ellet ole varma, ettei muokkaaminen vaikuta muihin kuviin.

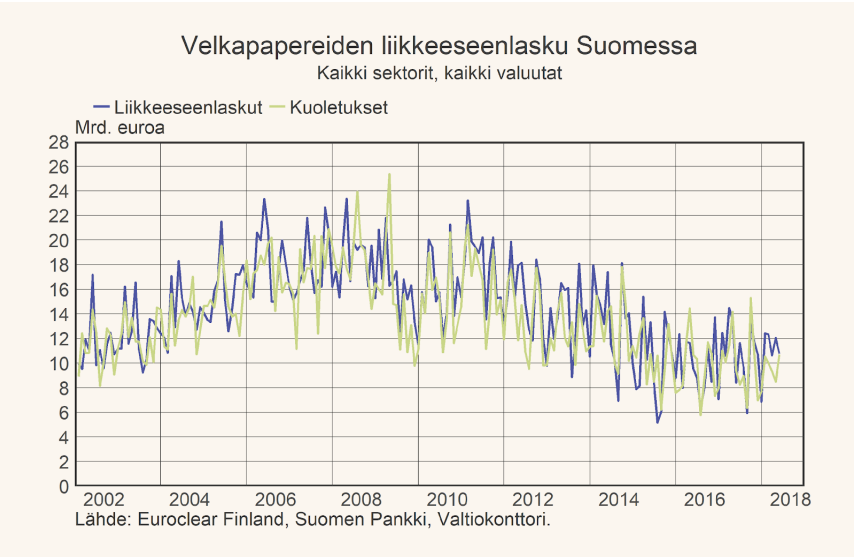
### 4.8 Lista alaohjelmista

Alla on lista tällä hetkellä käytössä olevista alaohjelmista, sekä kuvaukset niiden ominaisuuksista. Lista ohjelmista löytyy myös excel-taulukosta "Lista geom-tiedostoista". Päivitetty 13.7.2018.



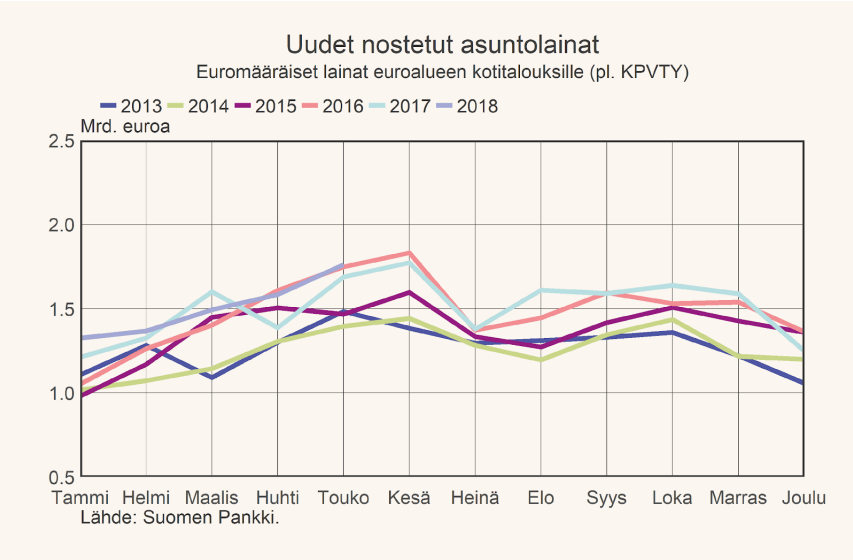

- **geom_line_longv4.R**

Piirtää viivakuvaajan, jossa **Y=Value, X=Date(Vuosi).** Käytettävän data framen nimi tulee olla **a_date,** muuttujan **SeriesName.**



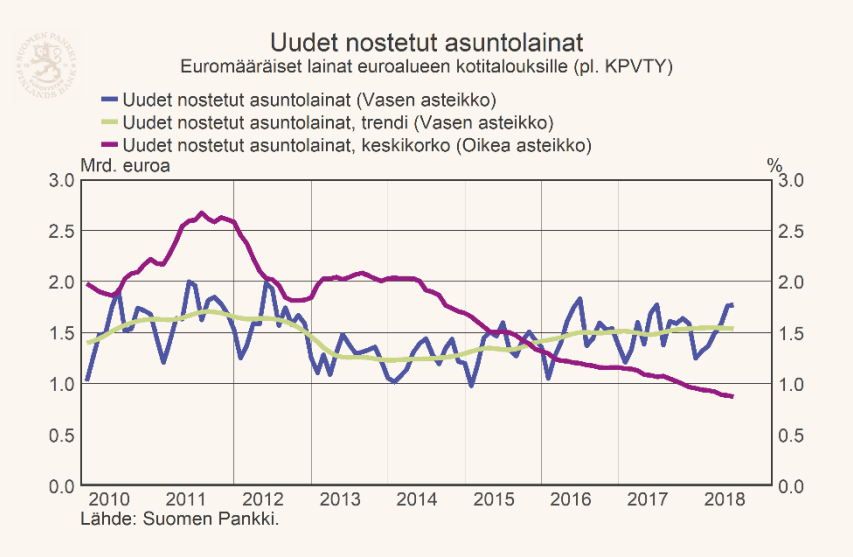

- **Geom_line_long_by_2_years.R**

Piirtää viivakuvaajan, jossa **Y=Value, X=Date(Joka toinen vuosi).** Käytettävän data framen nimi tulee olla **a_date,** muuttujan **SeriesName.**



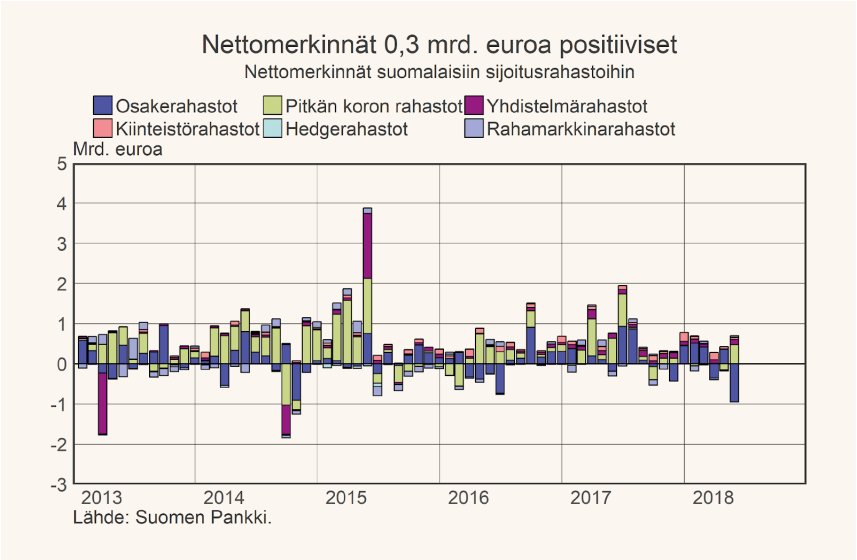

- **Geom_line_long_3_v2.R**

Piirtää viivakuvaajan, jossa **Y=Value, X=Date(Kuukausi).** Käytettävän data framen nimi tulee olla **a_date** muuttujana **variable(vuosi).**



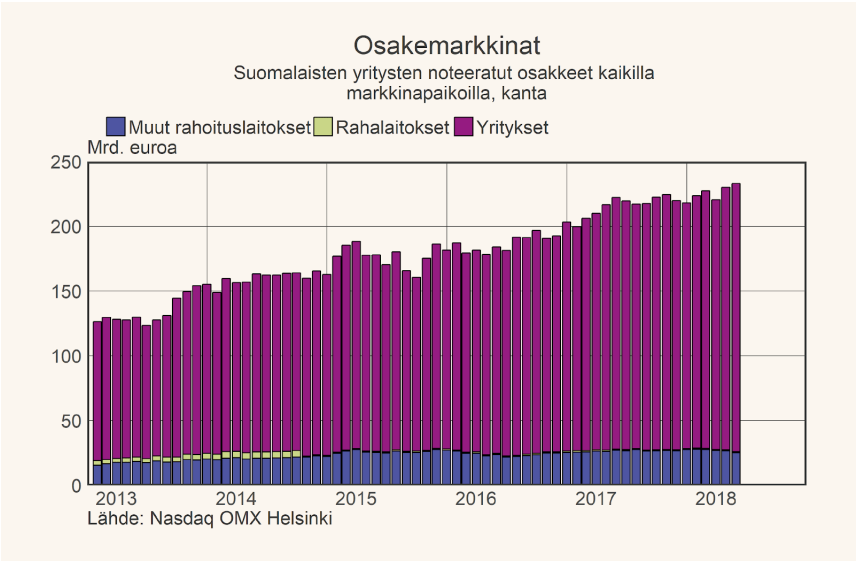

- **Geom_line_long_twoaxis.R**

Piirtää viivakuvaajan kahdella y-akselilla, jossa **Y=Value, X=Date(Vuosi).** Käytettävän data framen nimi tulee olla **a_date,** muuttujan **SeriesName.**



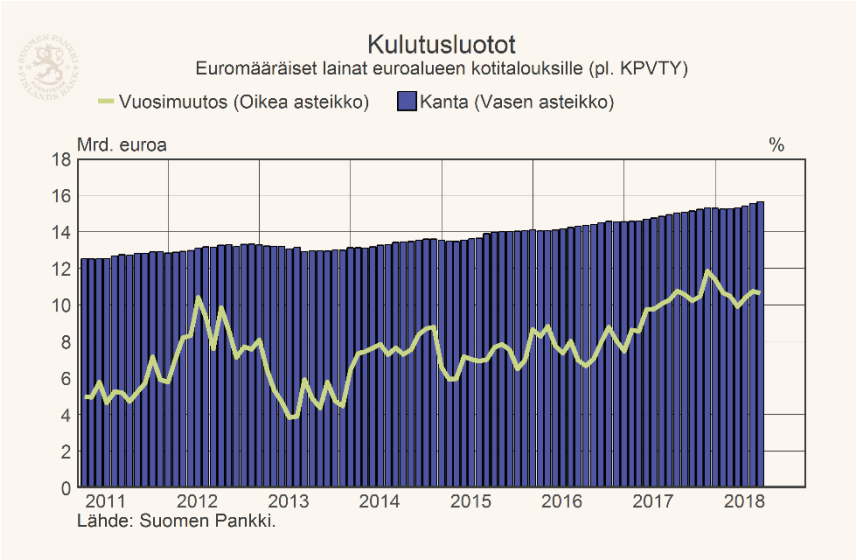

- **geom_bar_stack.R**

Piirtää ggplotin automaattisessa järjestyksessä päällekkäisen palkkikuvaajan, jossa **y=Value, X=Date(Vuosi).** Käytettävän data framen nimi tulee olla **a_date,** muuttujan **SeriesName.**




- **geom_bar_stack_order.R**

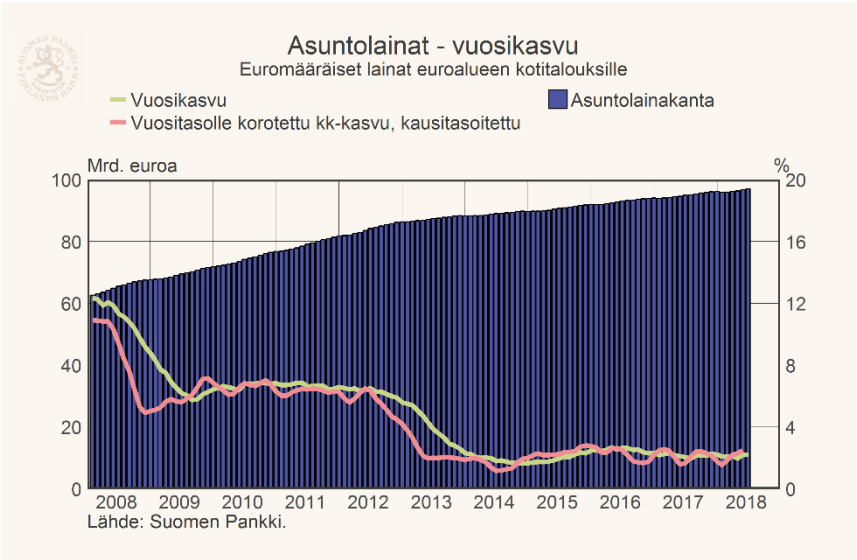
Piirtää itse määritellyssä järjestyksessä päällekkäisen palkkikuvaajan, jossa **y=Value, X=Date(Vuosi).** Käytettävän data framen nimi tulee olla **a_date,** muuttujan **variable.** Ohjeet järjestyksen määrittelemiseen kohdassa "4.6. Piirtojärjestyksen muuttaminen".




- **geom_bar_line.R**

Piirtää palkki- ja viivakuvaajan yhdistelmän, jossa **y=Value, X=Date(Vuosi).** Y-akseleita on kaksi, viivoja vain yksi. Tiedot otetaan eri data frameista:

**df1_date ->viiva df2_date->palkit**




- **geom_bar_line_2.R**

Piirtää palkki- ja viivakuvaajan yhdistelmän, jossa **y=Value, X=Date(Vuosi).** Y-akseleita ja viivoja on **kaksi**. Tiedot otetaan eri data frameista:

**df1_date ->palkit df2_date & df3_date->viivat**

Jakelu