1. Henkilötiedot

Numeerisen datan visualisointikirjasto, Helena Hilander, 655031, Bioinformaatioteknologia, 20.4.2019

2. Yleiskuvaus

- Ohjelmakirjasto, jolla voidaan visualisoida numeerista dataa graafisesti:

    - Datatiedosto luetaan käyttäjältä. Datan tulee noudattaa tiettyä formaattia, jotta ohjelma toimii (määritelty myöhemmin). Ohjelman mukana on esimerkkidatatiedostoja.

- Ohjelma piirtää perusviivadiagrammin:

    ● Piirtää 1-n käyrää, joilla jokaisella on oma numeerinen x-y data. Jokaisen kuvaajan omat datapisteet yhdistetään viivoilla. Käyttäjä voi myös poistaa datapisteiden väliset viivat, jolloin vain datapisteet näkyvät.

    ● Jokaisella käyrällä on oma selite, joka luetaan datasta. Eri käyrät piirretään eri väreillä ja jokaisen käyrän selite ja väri näkyvät ruudulla infolaatikossa.

    ● Käyttäjä voi antaa kuvaajalle otsikon ja akseleiden nimet.

    ● Käyttäjä voi lisätä gridin kuvaajan taakse ja ottaa sen pois.

- Työ on tehty keskivaikeana versiona alkuperäisestä suunnitelmasta poiketen.

3. Käyttöohje

Tallenna ensin esimerkkitiedostot data1.txt ja data2.txt koneelle. Ohjelma käynnistetään ajamalla mainwindow.py tiedosto. Seuraavaksi valitaan piirrettävä data menubarin ”file” valikon ”open file” painikkeella. Datatiedosto valitaan käyttäjän koneelta. Kun tiedosto on valittu, ohjelma piirtää kuvaajan automaattisesti. Mikäli data ei noudata ohjelman vaatimaa muotoilua, ohjelma ilmoittaa tästä virheenä.

Piirron onnistuttua käyttäjä voi muokata kuvaajaa ”Figure settings” -valikon toiminnoilla. Käyttäjä voi poistaa pisteiden väliset viivat, laittaa gridin päälle/pois, antaa kuvaajalle otsikon sekä akseleille nimet.

4. Ulkoiset kirjastot

Käytetyt ulkoiset kirjastot ja selitteet käytölle:

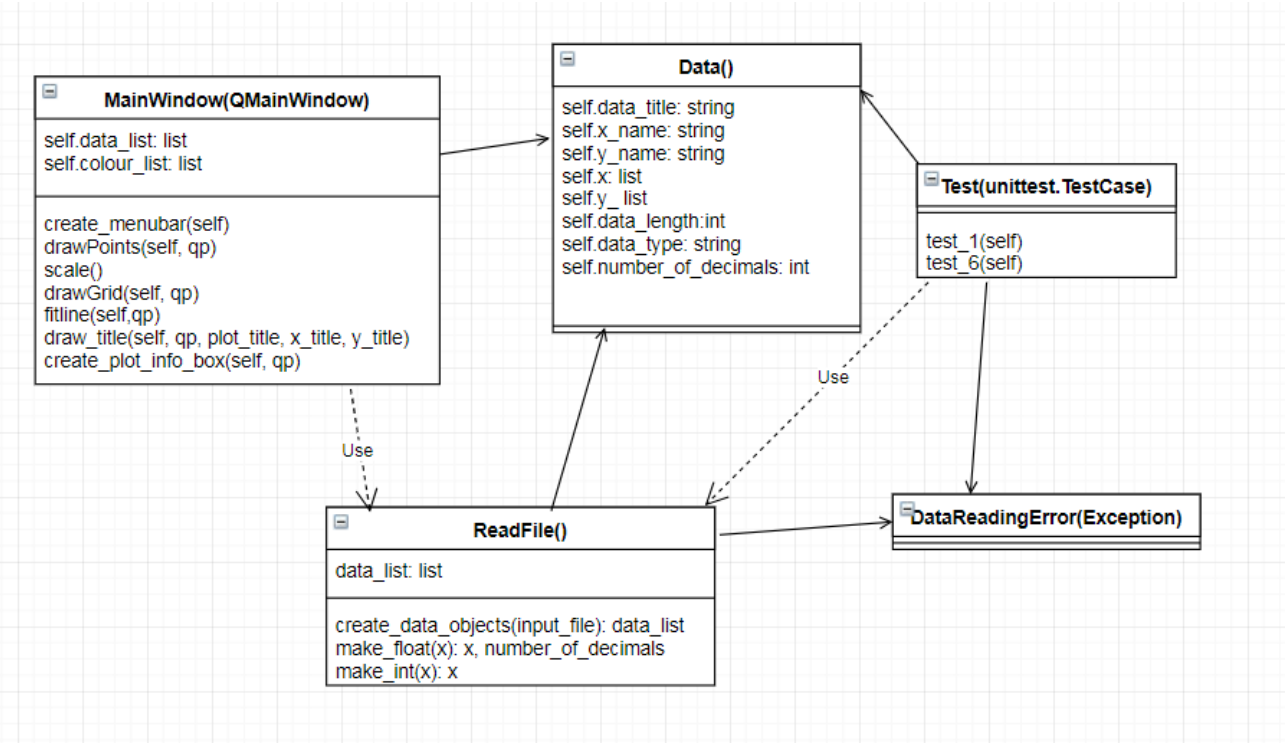
- PyQt5: Graafisen käyttöliittymän luonti.
- Math: Itseisarvojen laskeminen piirtoalgoritmeissa.
- Unittest: ReadFile() luokan testaaminen.

5. Ohjelman rakenne

Ohjelman luokkajako:

- **MainWindow(QMainWindow)**
    - Luo ohjelman graafisen käyttöliittymän
    - Hoitaa kaiken graafisen piirron sekä datan ja gridin skaalauksen näytölle sopiviksi.
- **Data():**
    - Data-luokan olio on yhden datan tietojen varasto. (Yhdessä tiedostossa voi olla useampi datasetti peräkkäin, joista jokaisella on oma Data-olio.)
    - Data -luokan oliolla on vain attribuutteja, ei metodeja.
- **ReadFile()**
    - Tiedosto luku, Data olioiden luominen ja kokoaminen listaksi
    - Mainwindow luokan metodi showDialog kutsuu Readfile luokan create_data_objects() metodia, joka luo Data-oliot tiedostosta, kokoaa ne listaksi, palauttaa listan Mainwindow:lle kuvaajan piirtoa varten.
    - Mikäli tiedoston muotoilu ei vastaa vaatimuksia, ReadFile() nostaa DataReadingErrorin
- **DataReadingError(Exception):**
    - Virheluokka, jonka tyypin virheen ReadFile() nostaa tiedoston luvun epäonnistuessa.
- **Test(unittest.TestCase):**
    - Yksikkötestausluokka ReadFile luokan testaukseen.
    - Testataan, että ReadFile luokka nostaa virheellisille tiedostoille DataReadingErrorin ja luo virheettömästä tiedostosta oikeanlaisen Data-olion.

UML-KAAVIO:


MainWindow(QMainWindow)
self.data_list: list
self.colour_list: list
create_menubar(self)
drawPoints(self, qp)
scale()
drawGrid(self, qp)
fitline(self,qp)
draw_title(self, qp, plot_title, x_title, y_title)
create_plot_info_box(self, qp)
Data()
self.data_title: string
self.x_name: string
self.y_name: string
self.x: list
self.y_ list
self.data_length:int
self.data_type: string
self.number_of_decimals: int
Test(unittest.TestCase)
test_1(self)
test_6(self)
Use
Use
ReadFile()
data_list: list
create_data_objects(input_file): data_list
make_float(x): x, number_of_decimals
make_int(x): x
DataReadingError(Exception)

6. Algoritmit

Skaalausalgoritmi:

- **Kuvaajan piirto perustuu datapisteiden välisten etäisyyksien skaalaamiseen käyttöliittymän ikkunan kokoon.**

(Selitettynä x koordinaatille. Sama algoritmi pätee y-koordinaatille, kun leveys korvataan korkeudella ja huomioidaan ero 5. kohdassa.)

1. Etsitään piirrettävästä datasta minimi- ja maksimiarvo
2. Lasketaan etäisyys minimi- ja maksimiarvon välillä: $etäisyys = |\max - \min|$
3. Selvitetään datapisteiden skaalauskerroin jakamalla ikkunan leveys datapisteiden etäisyydellä. Ikkunan leveydestä vähennetään pienet sivumarginaalit, jotta kuvaajan reunoille jäisi tilaa: $\text{skaalauskerroin} = (\text{ruudun leveys} - 2 * \text{marginaali})/\text{etäisyys}$
4. Määritellään normalisointivakio, siirtämään datan pienin arvo alkamaan nollasta: $if\ \min < 0 : normalisointivakio = |\min|, else\ normalisointivakio = -min$
5. Piirretään datapisteet seuraavanlaisessa muodossa:

$$x_{piirrettävä} = (x_{data} + normalisointivakio) * skaalauskerroin + marginaali$$

Y-koordinaatin piirto toteutetaan hieman eri lailla. Tämä johtuu Qt:n koordinaattijärjestelmästä: 0,0 koordinaatti on vasemmassa yläkulmassa. Tähän ohjelmaan haluamme 0,0 koordinaatin vasempaan alakulmaan, joten joudumme ”kääntämään” y-koordinaatit siten, että datassa suurin y-koordinaatti saa piirrettäessä pienimmän arvon, ja piirtyy vasempaan yläkulmaan.

Y koordinaatin piirto:

$$y_{piirrettävä} = korkeus_{ikkuna} - marginaali - (y_{data} + normalisointivakio) * skaalauskerroin$$

- Perustelu skaalausalgoritmille: Datapisteet sijoittuvat siististi näytölle säilyttäen samat suhteelliset etäisyydet kuin alkuperäisessä datassa. Suhteellinen laskentatapa mukautuu ikkunan koon muutoksiin toisin kuin suoraan piirto datapisteiden alkuperäisillä koordinaateilla.

Gridin piirto -algoritmi:

- **Gridin piirto perustuu vaaka ja pystyviivojen piirtoon ikkunan kokoon suhteutetun etäisyyden välein.**

1. Valitaan, kuinka tiheä grid-verkko halutaan (= kuinka monta pienempää neliötä vaaka/pystysuunnassa verkossa on). Alkuperäisessä koodissa arvoksi on valittu 5: $lkm_{neliöt} = 5$
2. Lasketaan piirrettävän pienen neliön sivun pituus jakamalla kuvaajan leveys neliöiden lukumäärällä (etäisyys_x = minimi- ja maksimiarvon välinen etäisyys datassa. Kun etäisyys_x kerrotaan skaalauskertoimella, saadaan datan näytöllä viemä tila vaakasuunnassa):

   $koko_{neliö} = etäisyys_x * skaalauskerroin\ /lkm\_neliöt$
3. Piirretään vaaka ja pystysuuntaisia viivoja aina neliön sivun pituuden välein. Alla annettu vaakaviivojen y-koordinaatin sekä pystyviivojen x-koordinaatin kaava:

   $i = [0, lkm_{neliöt}]$

   $vaakaviiva_y = korkeus_{ikkuna} - marginaali - (y_{max} + normalisointivakio) * skaalauskerroin + koko_{neliö} * i$

   $$pystyviiva_x = (x_{min} + normalisointivakio) * skaalauskerroin + marginaali + koko_{neliö} * i$$

- Perustelu gridin piirto -algoritmin valinnalle: Suhteelliset etäisyydet mukautuvat ikkunan koon vaihteluun.

7. Tietorakenteet

Ohjelmassa tärkeä varastoitava ja käsiteltävä tieto on tiedostosta luettu data. Jokainen yksittäinen datasetti esitetään ohjelmassa Data-oliona, ja oliot kootaan listaksi: Mainwindow luokan metodi showDialog() kutsuu Readfile-luokan create_data_objects() metodia, joka luo Data-oliot tiedostosta, kokoaa ne listaksi, palauttaa listan Mainwindow:n attribuutiksi.

Data -oliolla on vain attribuutteja, eikä ollenkaan metodeja, eli sitä käytetään vain tiedon varastoimiseen.

8. Tiedostot

- Ohjelma tarvitsee toimiakseen oikein muotoillun tekstitiedoston. Tiedoston tulee noudattaa seuraavaa muotoilua:

  #data1_title
  #x1_name ,y1_name
  #x1_type ,y1_type
  x0 , y0
  xn , yn
  #data2_title
  #x2_name ,y2_name
  #x2_type ,y2_type
  x0 , y0
  xn , yn

- Mikäli haluttaisiin piirtää yksiulotteista dataa (ei vielä mahdollista ohjelmalla), Tiedostosta voidaan jättää y-koordinaattien tiedot ja pilkut pois.
- Yhdessä tiedostossa voi olla useita (myös enemmän kuin kaksi) datasettiä, kunhan kaikilla dataseteillä on alussa #-merkityt tiedot oikein.
- Datasettien tulee olla täysin peräkkäin tiedostossa, eikä niiden välillä saa olla välilyöntiä.
- x_type/y_type on joko int, float tai string.
- Tällä hetkellä ohjelma piirtää ainostaan int ja float muotoista dataa kahdella koordinaatilla. Tiedoston muoto ja datan luku on kuitenkin tehty siten, että laajennus ei-numeerisen ja yksiulotteisen datan piirtämiseen olisi mahdollista, kuten projektin vaikeassa versiossa vaaditaan.
- Liitteenä kaksi oikein muotoiltua tekstitiedostoa data1.txt ja data2.txt ohjelman testaamiseen. Tiedostot tulee tallentaa koneelle ja lukea käyttöliittymistä ”read file” komennolla.

9. Testaus

Ohjelmaa testattiin jo suunnitteluvaiheessa, ja ReadFile-luokan testaamiseksi luotiin yksikkötestausluokka.

Luokka Test(unittest.TestCase) sisältää kuusi kappaletta ReadFile luokan testaamiseen tehtyjä yksikkötestejä. Viisi ensimmäistä testiä varmistavat, nostaako ReadFile DataReadingErrorin erilaisten virheellisten tiedostojen yhteydessä. Kuudes testi testaa, että ReadFile:n luomalla Data-objektilla on virheettömän tiedoston mukaiset attribuutit, eli, että Data olio luodaan oikein. ReadFile läpäisee kaikki kuusi testiä.

Graafisesta käyttöliittymästä huolehtivaa MainWindow-luokkaa on testattu erilaisilla datatiedostoilla, ja akselien numerointi, kuvaajien muoto ovat säilyneet oikeina, eikä huomattavaa akselien ja kuvaajan päällekkäisyyttä esiinny, ellei ikkunasta tehdä todella pientä.

10. Ohjelman tunnetut puutteet ja viat

- Tiedoston muotoiluformaatti on aika rajoittava. Tiedoston muotoilussa havaittu vika on, ettei kahden eri datasetin välissä voi olla välilyöntiä. Ohjelma kaatuu, mikäli tiedoston keskellä on välilyönti.
- Kun luetaan uusi tiedosto, vanhan tiedoston otsikko ja akseleiden nimet jäävät näkyviin. Tämän voisi muuttaa laittamalla draw_title() funktion kutsun alustavilla nimillä drawPoints() funktioon.

11. 3 parasta ja 3 heikointa kohtaa

Parhaat asiat:

1. Suhteellisesti ikkunan kokoon mukautuva kuvaajan piirto.
2. Yksinkertainen ja simppeli käyttöliittymän rakenne (hyvässä ja pahassa..)
3. Randomisoitu värivalinta, jotta jokainen kuvaaja saadaan piirrettyä eri väreillä

Heikoimmat kohdat:

1. Tylsä ja alkeellinen ulkonäkö: Päätin, että käytän ensisijaisesti aikaa mahdollisimman virheettömän ohjelman rakentamiseen, joten aikaa ei lopulta jäänyt ulkonäön miettimiseen. Lähtisin seuraavaksi kehittämään tätä.

2. Kaikki graafinen piirto yhdessä MainWindow luokassa: Mikäli ohjelmaa jatketaan vaikeaan versioon, ja eri kuvaajatyyppejä lisätään, tulee MainWindow luokasta hyvin pitkä ja monimutkainen. Tämän voisi ratkaista jakamalla jokaisen eri kuvaajatyypin piirron omaksi luokakseen. Jokaisen kuvaajatyypin luokka loisi uuteen ikkunaan avautuvan QWidget() objektin ja piirtäisi siihen kuvaajan. (Tämä oli alkuperäinen suunnitelma, mutta aloitin testailun tekemällä kaiken pääikkunaan, enkä enää ehtinyt muuttamaan järjestelyä.)

12. Poikkeamat suunnitelmasta

Alkuperäinen suunnitelma oli toteuttaa projektin vaikea versio useilla eri kuvaajatyypeillä, mutta muut kurssit ja työt veivät enemmän aikaa kuin olin arvioinut, joten en ehtinyt tehdä vaativaa versiota loppuun.

Aloitin kuitenkin projektin vaativa versio mielessä, ja tämä näkyy esimerkiksi Data-olion attribuuteista: data_type attribuutti on tarkoitettu määrittelemään, minkä tyyppistä data on, ja mitä kaikkia kuvaajia siitä voitaisiin piirtää. (Esim data_type = singlestring tarkoittaa, että data on yksiuloitteinen merkkijonodata, jolloin siitä voidaan piirtää PieChart tai BarPlot mutta ei viivadiagrammia.) Data_type määritellään #x_type ,y_type -tiedostorivin avulla.

13. Toteutunut työjärjestys ja aikataulu

ReadFile koodaus, Data-luokan koodaus	viikko 9 (~10h)
Test()-luokan koodaus ja dokumentaation luku	viikko 10 (~10h)
PyQt5 dokumentaation luku	viikko 11 (~10h
Mainwindow koodaus (pisteiden piirto ja grid)	viikko 12 (~10h)
Mainwindow koodaus (viivojen piirto ja gridin parantelu)	viikko 13 (~10h)
Mainwindow koodaus (akseleiden ja kuvaajan nimeäminen, infolaatikko)	viikko 14 (~10h)
Dokumentaation kirjoitus ja koodin siistiminen	viikko 15 (~10h)
Dokumentaation viimeistely, mainwindow:n viimeinen siistiminen ja testaus	viikko 16 (~10h)

14. Arvio lopputuloksesta

Numeerisen datan visualisointikirjasto -projektin lopputulos on yksinkertainen, siististi toimiva ohjelma. Olen tyytyväinen lopputulokseen, sillä olen ensimmäistä kertaa opiskellut graafisen käyttöliittymän koodaamista tällä kurssilla, ja projekti oli ensimmäinen kokonainen ohjelma, jonka olen itse rakentanut alusta loppuun.

Ohjelmassa on paljon kehitettävää, ja pyrin kommentoimaan koodin siten, että ohjelman jatkotyöstäminen olisi helppoa myös sen jälkeen, kun kaikki projektiin liittyvä on päässyt unohtumaan.

Lähtisin seuraavaksi työstämään visuaalista puolta: muuttaisin mainwindow:n yleisulkonäköä pois default-muotoilusta.

Jatkaisin myös projektin vaikean version työstämistä ja jakaisin mainwindow:n useampaan alaluokkaan, ja piirtäisin kuvaajat erilliseen ikkunaan. Ohjelman laajentaminen olisi helppoa, sillä datan luku ja tallennus on tehty vaativampaa versiota varten.

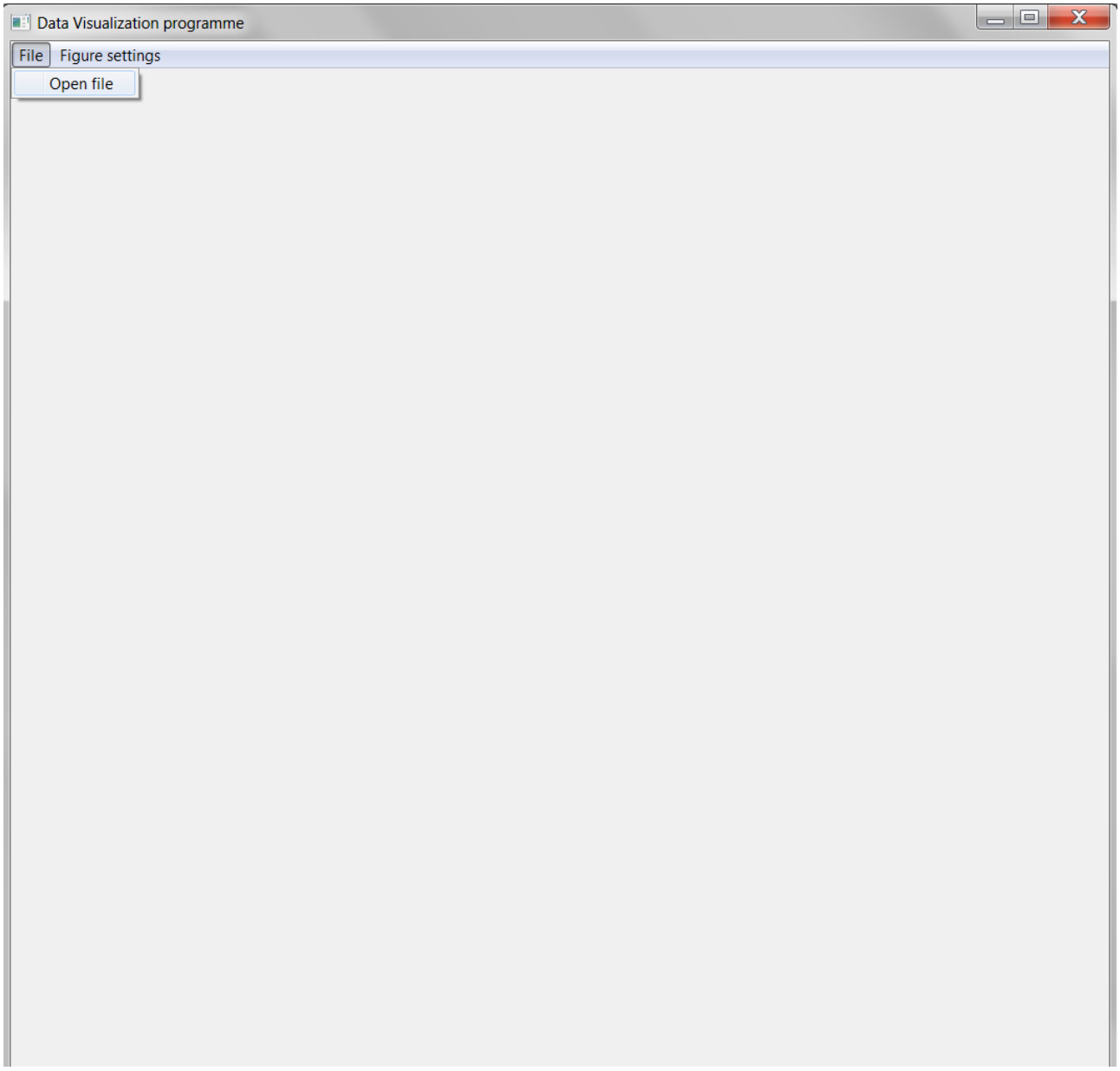
Lähtisin myös selvittämään, miten tiedoston muotoilusta voisi saada vähemmän rajoittuneen. (Nyt ohjelma kaatuu, jos tiedoston keskellä on väli.Tämän voisi ehkä estää esim if(line == ””: pass -käskyllä.)

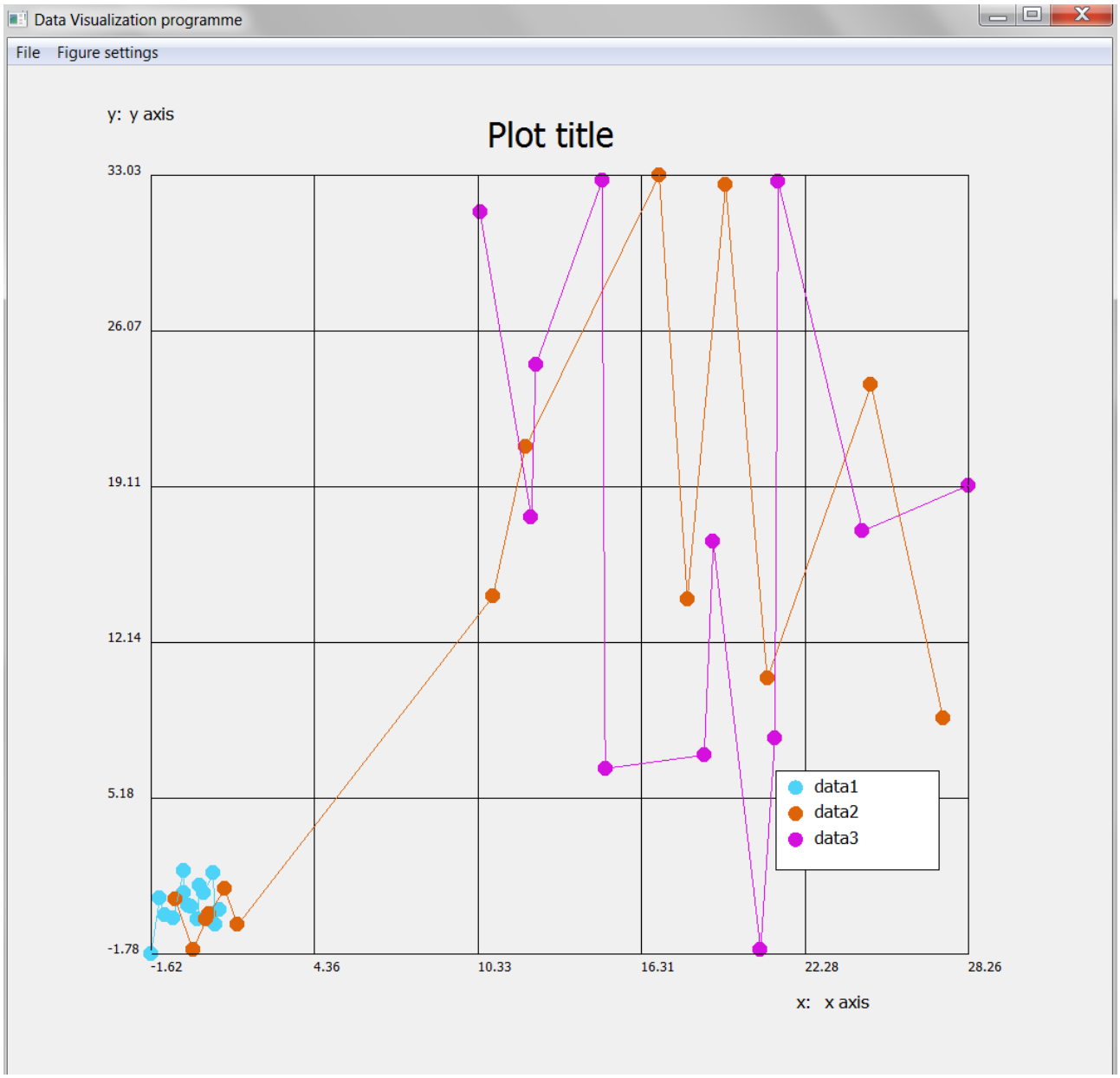
15. Viitteet

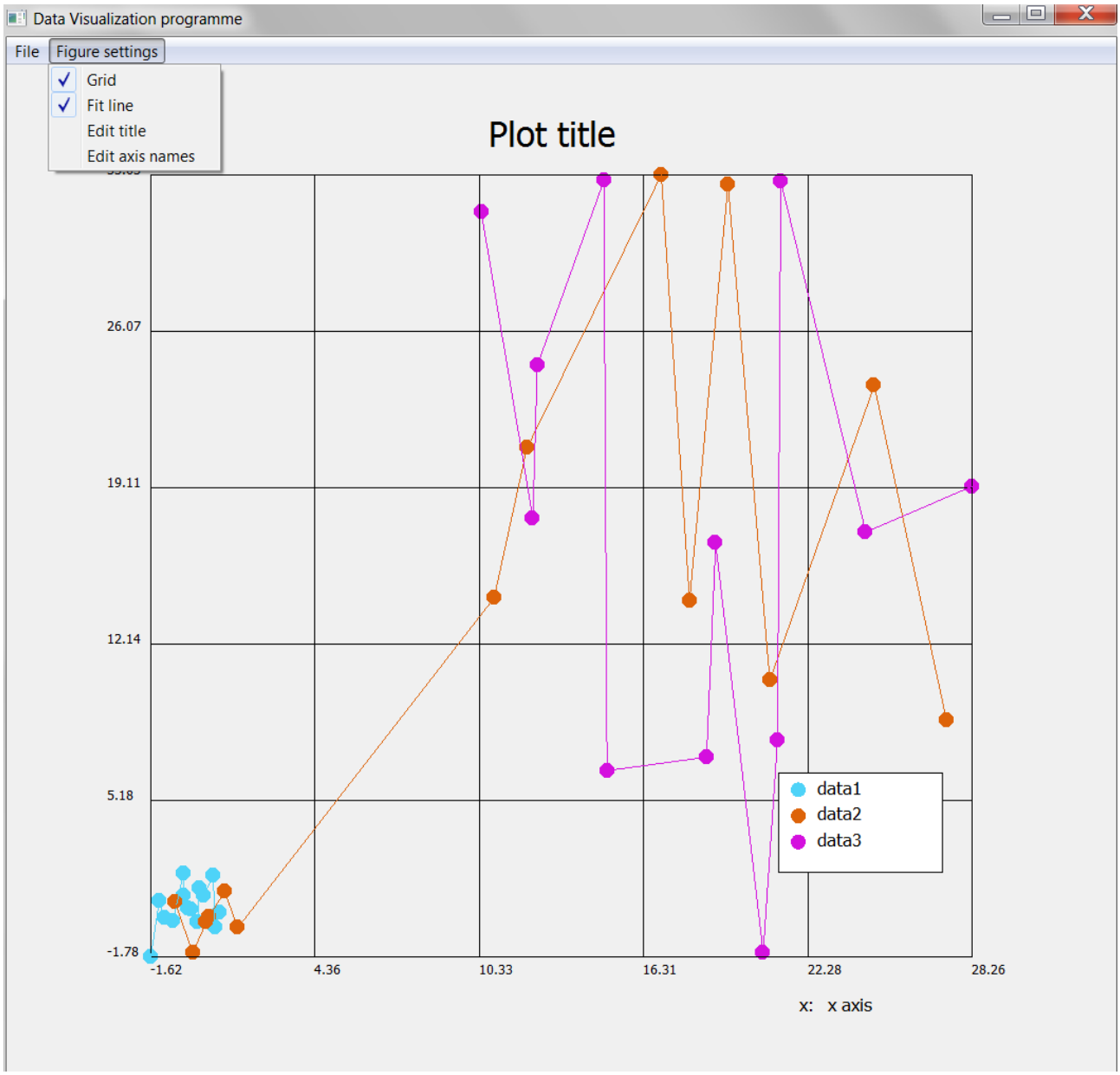
- https://www.zeolearn.com/magazine/10-steps-for-getting-started-guis-with-python 20.2.2019
- https://data-flair.training/blogs/python-pyqt5-tutorial/ 20.2.2019
- https://pynative.com/python-check-user-input-is-number-or-string/ 21.2.2019
- https://www.qt.io/ 23.2.2019
- http://zetcode.com/gui/pyqt5/ 22.2.2019
- https://docs.python.org/3/ 23.2.2019

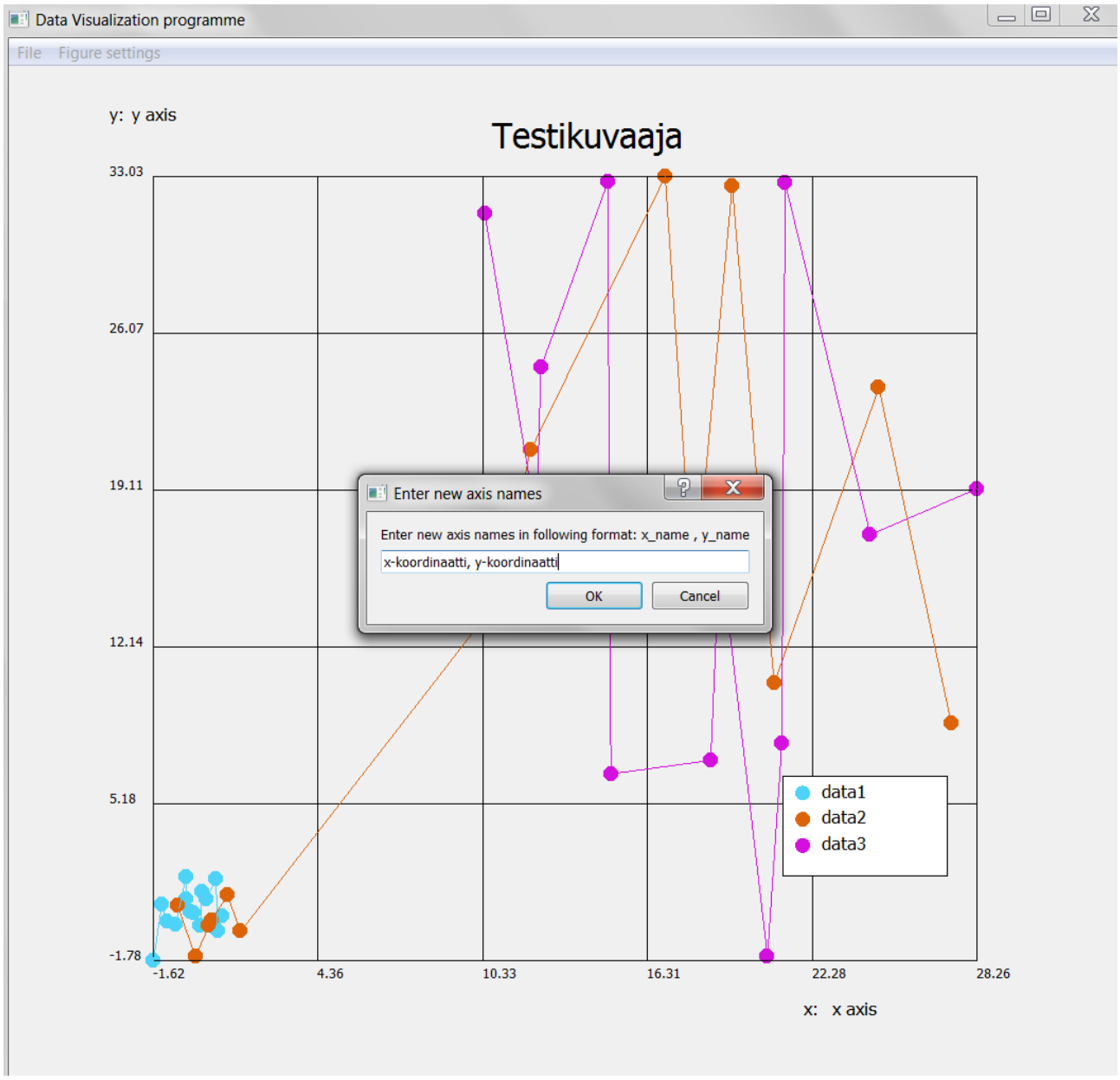
16. Liitteet

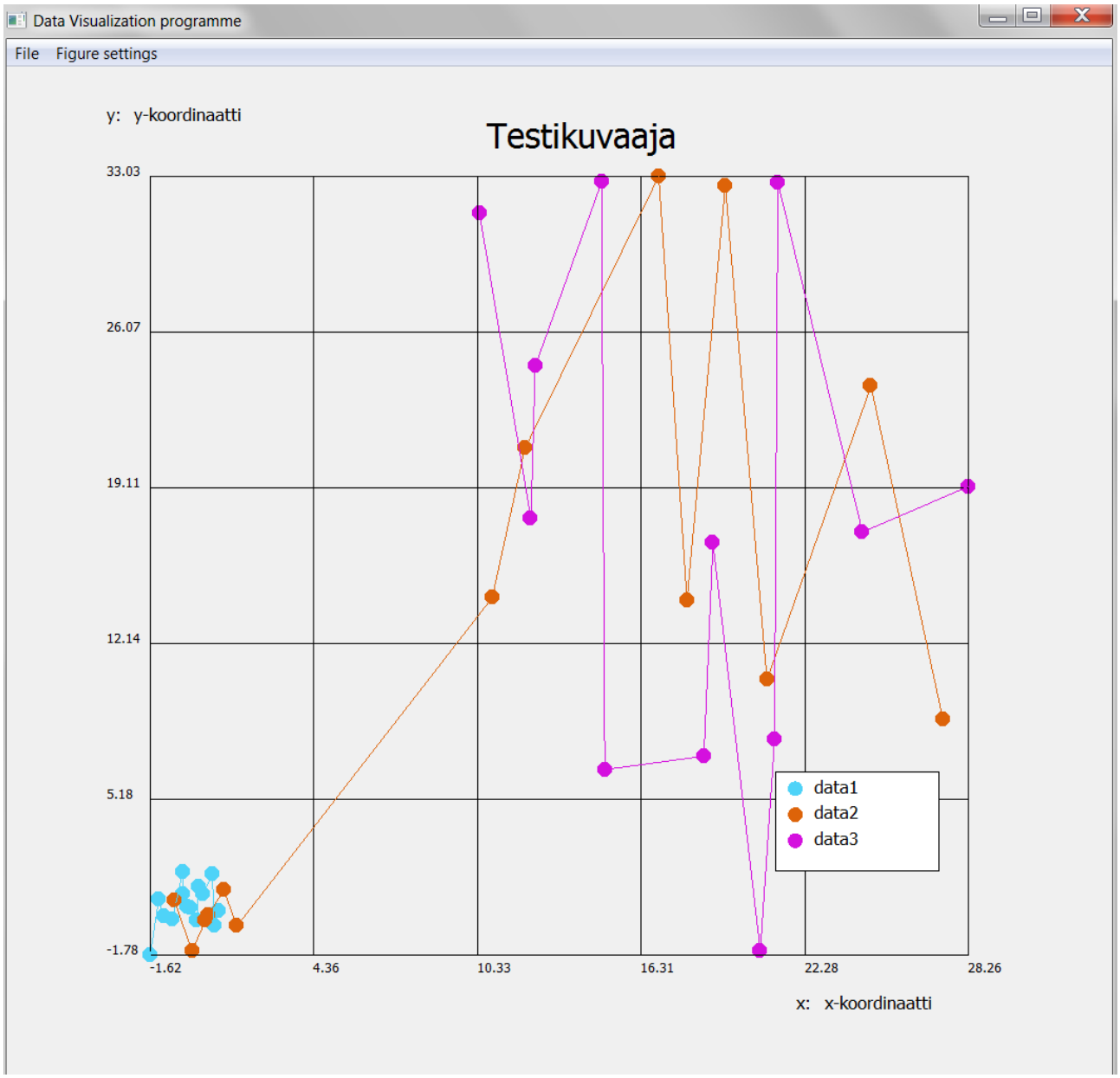
Käyttöliittymäkuvia:




Data Visualization programme
File
Figure settings
y: y axis
Plot title
33.03
26.07
19.11
12.14
5.18
-1.78
-1.62
4.36
10.33
16.31
22.28
28.26
x: x axis
data1
data2
data3


Data Visualization programme
File
Figure settings
Grid
Fit line
Edit title
Edit axis names
Plot title
26.07
19.11
12.14
5.18
-1.78
-1.62
4.36
10.33
16.31
22.28
28.26
x: x axis
data1
data2
data3


Data Visualization programme
File
Figure settings
y: y axis
Testikuvaaja
33.03
26.07
19.11
12.14
5.18
-1.78
-1.62
4.36
10.33
16.31
22.28
28.26
x: x axis
data1
data2
data3
Enter new axis names
Enter new axis names in following format: x_name , y_name
x-koordinaatti, y-koordinaatti
OK
Cancel


Data Visualization programme
File
Figure settings
y: y-koordinaatti
Testikuvaaja
33.03
26.07
19.11
12.14
5.18
-1.78
-1.62
4.36
10.33
16.31
22.28
28.26
data1
data2
data3
x: x-koordinaatti