Numeerisen data visualisointi

Tekninen suunnitelma

Anton Mattsson

426370

BioIT 3

Ohjelman rakennesuunnitelma

Ohjelman käyttämiä luokkia on kuvattu seuraavan sivun UML-kaaviossa. En ole vielä selvittänyt tarkalleen, mitä PyQt:n luokkia ohjelmani luokat perivät, joten niitä ei ole kuvattu tässä suunnitelmassa.

DataReader luokka lukee tiedot käyttäjän antamasta .csv-tiedostosta ja palauttaa arvot sanakirjana.

Kuvaajan pohjana toimii Figure-luokka, joka luo tyhjän ikkunan ja varaa siihen tilaa itse kuvaajalle sekä akselien ja kuvaajan nimille. Figure-luokalla on add_title-metodi, joka lisää kuvaajaan otsikon sekä set_margins-metodi, jolla eri kuvaajien luokat voivat varata tarvitsemilleen nimille tilaa. Otsikkoa kuvaa FigureTitle-luokka.

Plot on yläluokka, jonka kaikki eri kuvaajatyypit perivät. Tässä työssä toteutetaan vain aliluokka LinePlot, mutta uusia kuvaajia voisi helposti lisätä. Plot-luokalla on metodi draw_plot, joka piirtää kuvaajan.

LinePlot perii Plot-luokan ja kuvaa viivadiagrammia. LinePlotin metodit ovat:

- calculate_transformation: muuntaa datan arvot pikseleiksi piirtämistä varten. Tallentaa muunnoksen kertoimet.
- draw_plot: suorittaa muut piirtometodit
- draw_axes: piirtää kuvaajan akselit
- draw_curves: piirtää kuvaajan käyrät
- set_grid_origin: asettaa kuvaajan origon gridin piirtämistä varten
- draw_grid: piirtää gridin
- draw_legend: lisää kuvaajaan selitteen

Plot-oliot sisältävät aina monta PlotPart-oliota. Käytännössä jokainen kuvaajan osa, joka tarvitsee oman maininnan selitteessä, on PlotPart-olio. Esimerkiksi LinePlot-luokka käyttää piirtämiseen PlotPart-luokan aliluokkaa Curve, joka kuvaa yhtä viivadiagrammin käyrää (murtoviivaa). Curve taas piirtää useamman Line-olion, joka kuvaa kahta pistettä yhdistävää viivaa.

Legend-luokka pitää huolta kuvaajan selitteen piirtämisestä. Legendiin kuuluu monta LegendPart-oliota, joista kukin piirtää yhtä PlotPart-oliota (Curve) vastaavan osan selitteestä, eli tässä tapauksessa käyrän värin ja nimen.

Kuvaajan akseleita varten on olemassa Axis-luokka. Axis käyttää hyväkseen LinePlotin tekemän muunnoksen arvoja ja piirtää akselien pisteet oikeisiin kohtiin ja nimeää ne. Jokainen akselille merkitty viiva on AxisTick-olio, eli pysty- tai vaakasuora viiva, johon liittyy sijainti akselilla ja arvo. Lisäksi akseleilla on oma nimi ja myös asteikon voi nimetä.

Gridin piirtämistä varten on Grid-luokka, jonka draw_grid_lines metodi piirtää gridin muodostavat viivat. Jokaista viivaa varten luodaan Gridline-olio, joka piirtää vaaleanharmaan pysty-tai vaakasuoran viivan. Myös Grid-luokka käyttää hyväkseen LinePlotin tekemää muunnosta datan arvojen ja pikselien välillä.

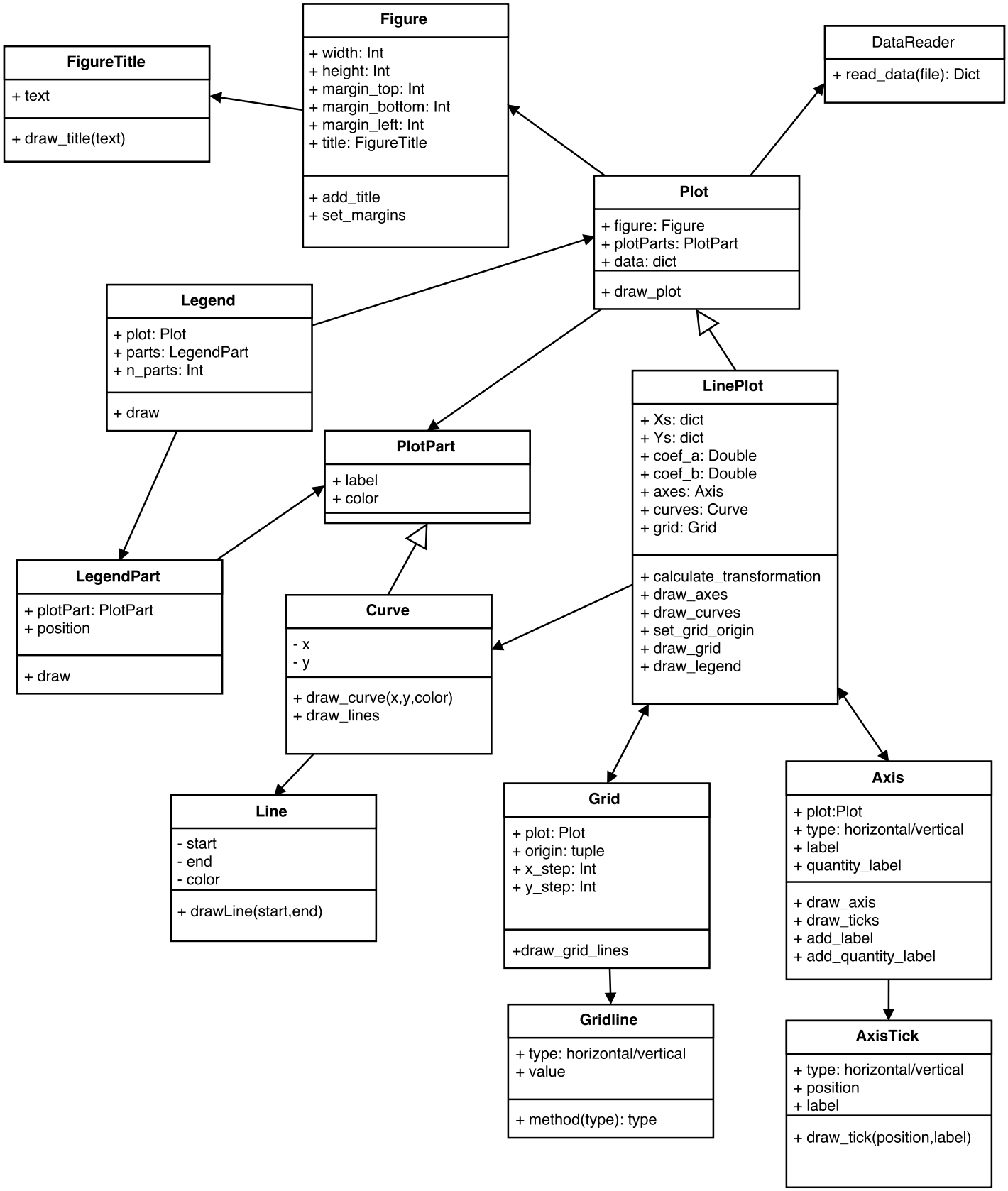
Figure
+ width: Int
+ height: Int
+ margin_top: Int
+ margin_bottom: Int
+ margin_left: Int
+ title: FigureTitle
+ add_title
+ set_margins
FigureTitle
+ text
+ draw_title(text)
DataReader
+ read_data(file): Dict
Plot
+ figure: Figure
+ plotParts: PlotPart
+ data: dict
+ draw_plot
Legend
+ plot: Plot
+ parts: LegendPart
+ n_parts: Int
+ draw
PlotPart
+ label
+ color
LinePlot
+ Xs: dict
+ Ys: dict
+ coef_a: Double
+ coef_b: Double
+ axes: Axis
+ curves: Curve
+ grid: Grid
+ calculate_transformation
+ draw_axes
+ draw_curves
+ set_grid_origin
+ draw_grid
+ draw_legend
LegendPart
+ plotPart: PlotPart
+ position
+ draw
Curve
- x
- y
+ draw_curve(x,y,color)
+ draw_lines
Line
- start
- end
- color
+ drawLine(start,end)
Grid
+ plot: Plot
+ origin: tuple
+ x_step: Int
+ y_step: Int
+draw_grid_lines
Gridline
+ type: horizontal/vertical
+ value
+ method(type): type
Axis
+ plot:Plot
+ type: horizontal/vertical
+ label
+ quantity_label
+ draw_axis
+ draw_ticks
+ add_label
+ add_quantity_label
AxisTick
+ type: horizontal/vertical
+ position
+ label
+ draw_tick(position,label)

Käyttötapauskuvaus

Bioinformaatikko haluaa selvittää, miten kolme eri lääkettä vaikuttaa potilaiden verenpaineeseen ajan funktiona. Potilaiden verenpaineet ja mittausajat on tallennettu .csv-tiedostoon. Bioinformaatikko lataa datan muistiin ja syöttää sen piirtokomennolle. Piirtokomennolle hän antaa lisäksi tiedot siitä, mitkä sarakkeet hän haluaapiirtää vastakkain. Lisäksi hän kertoo, että akselien nimiksi tulisi asettaa time ja BP, sekä selitteen arvoiksi tulisi asettaa drug1, drug2 ja drug3. Kuvaajan otsikoksi hän asettaa Effect of tested drug to BP. Ohjelma piirtää hienon kuvaajan.

Algoritmit

Ainoa ohjelman käyttämä algoritmi on datapisteiden arvojen muuntaminen pikseleiksi piirtämistä varten. X-akselin arvoille kaava on muotoa:

$$X_{pixels} = (X_{raw} - min(X_{raw})) * \frac{Figure.width}{max(X_{raw}} \quad (1)$$

Y-akselin arvoille vastaavasti, mutta käytetään kuvaajan korkeutta.

Tietorakenteet

Luettu data tallennetaan sanakirjaan (dict), jonka arvot ovat monikkoja(tuple). Sanakirjan hyvä ominaisuus on, että muuttujan nimi kulkee kätevästi mukana avaimena. Arvot voi hyvin tallentaa staattiseen tietorakenteeseen, sillä pikseleiksi muutetut arvot talleneetaan erikseen, eikä arvoja tarvitse muuten muokata.

Aikataulu

IV-periodissa minulla on paljon muita kiireitä, joten projekti alkanee melko hitaasti.Tavoitteeni on toteuttaa checkpointiin mennessä seuraavat asiat:

- datan lataus: Tämän pitäisi olla muutamassa tunnissa valmis ja testattu
- Syvempi tutustuminen PyQt-kirjastoon: 7h
- Figure-luokan toteutus ja otsikon lisäys: 6h

Heti IV-periodin loputtua minulle jää enemmän aikaa työstää projektia. Tässä hahmottelemani työjärjestys:

- Yhden käyrän piirtäminen kuvaajaan, pisteiden skaalaus oikein, myös negatiivisia arvoja 20h
- Akselien lisäys 8h
- Gridin lisäys 6h
- Useampi käyrä ja selite 10h
- Testaus usealla eri käyrällä 5h
- Esimerkkikäyrän piirto 1h

Yksikkötestaussuunnitelma

Datan lataaminen on muusta ohjelmasta irrallinen osa, jota on helppo testata. Samoin pisteiden arvojen muuntaminen pikseleiksi pitäisi olla helppo omminaisuus testata. Muunnoksen yhteydessä tulee heti testata toimintaa useamman käyrän tapauksessa. Tällöin luonnollisesti minimi- ja maksimiarvoiksi valitaan kaikkien pisteiden minimi ja maksimi.

Muiden metodien testaus tehdään graafista kuvaa hyväksi käyttäen. Tämän takia luokka Figure toteutetaan ensin, jotta sen päälle rakentuvia metodeja on helppo testata visuaalisesti. Kuvasta näkee nopeasti, tuottaako metodi halutun tuloksen. Testausta varten tehdään valmiiksi muutamia datasettejä.

Kirjallisuusviitteet ja linkit

Työn pääasiallisena kirjallisuuslähteenä toimii PyQt:n ja Qt:n dokumentaatio: `http://pyqt.sourceforge.net/Docs/PyQt5/index.html` ja `http://doc.qt.io/qt-5/`. Lisäksi käytetään apuna erilaisia tutorialeja, joita listataan käytön mukaan.