Numeerisen data visualisointi

Projektidokumentti

Anton Mattsson

426370

BioIT 3

Yleiskuvaus

Työssä luotiin numeerisen datan visualisointikirjasto, jolla voi piirtää viivadiagrammeja. Muita kuvaajatyppejä voidaan helposti lisätä, eli työ on toteutettu keskivaikealla vaikeustasolla. Kuvaajassa annetun datan pisteet on yhdistetty viivoilla. Kuvaajaan kuuluvat myös otsikko, akselien nimet sekä selite, jotka ovat kaikki muokattavissa.

Visualisointikirjaston käyttöä varten luotiin esimerkkisovellus, jossa käyttäjä voi tarkastella satunnaiskävelysimulaation tuloksia.

Käyttöohje

Visualisointikirjastoa voi käyttää pythonin komentoriviltä tai osana lyhyttä ohjelmaa command_line.py-tiedostossa määritellyn line_plot-funktion avulla:

```
line_plot(file,
          pairs,
          title = 'Plot',
          xlabel = 'x axis',
          ylabel = 'y axis',
          curve_labels = None,
          gridon=True)
```

parametrien kuvaukset alla:

- file: merkkijono, polku tiedostoon, josta data luetaan
- pairs: kaksiulotteinen lista, alkiot ovat muuttujapareja, jotka tulee piirtää vastakkain
- title: merkkijono, kuvaajan otsikko
- xlabel: merkkijono, x-akselin nimi
- ylabel: merkkijono, y-akselin nimi
- curve_labels: lista merkkijonoja, käyrien nimet selitettä varten. Pituuden tulee olla sama kuin pairsin pituus. Oletuksena käytetään y-akselin muuttujien nimiä.
- gridon: totuusarvo, säätää gridin päälle tai pois päältä

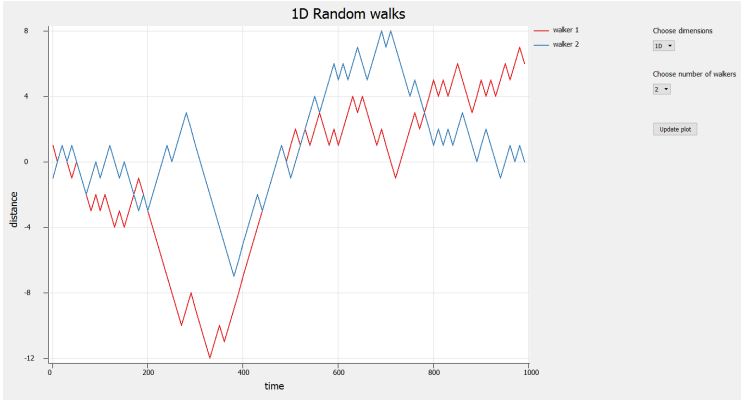
Esimerkiksi seuraava syöte:

```
line_plot('../data/rw2d.csv',
              [['x1','y1'],['x2','y2'],['x3','y3']],
              'Random walk 2D',
              'x',
              'y',
              curve_labels=['walker 1','walker 2','walker 3'])
```

Tuottaa seuraavan kuvaajan:

Kuvaajia voi myös sisällyttää osaksi laajempaa PyQt-sovellusta käyttämällä suoraan kirjaston luokkia (kuvattu seuraavassa osiossa).

Esimerkkisovellusta voi käyttää ajamalla main.py-tiedoston. Sovelluksessa voi valita yksi- ja kaksiulotteisen satunnaiskävelyn väliltä sekä valita kuvaajaan 1-3 satunnaiskävelijää. Alla on kuvankaappaus esimerkkisovelluksesta.



Ohjelman rakenne

Visualisointikirjaston käyttämiä luokkia on kuvattu seuraavan sivun UML-kaaviossa.

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 (width ja height) sekä akselien ja kuvaajan nimille ja selitteelle (margin_*)

Coordinates on koordinaattiluokka, jonka kentät kuvaavat pisteen sijaintia pikseleinä.

Plot on yläluokka, jonka kaikki eri kuvaajatyypit perivät. Tässä työssä toteutetaan vain aliluokka LinePlot, mutta uusia kuvaajia voisi helposti lisätä.

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).

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

- calculate_ticks(min_x, max_x): palauttaa 6-7 arvoa, jotka ovat sopivat kohdat kirjoittaa arvot akseleille
- calculate_coordinates: Muuntaa datan pisteet sekä calculate_ticks-metodin laskemat pisteet Coordinates-olioiksi
- add_curves: Luo piirtämistä varten jokaista käyrää varten Curve-olion

ColorMap käyttää `colorbrewer2.org`-sivuston värikarttoja ja luo kuvaajaa varten käyrien lukumäärän verran toisitaan erottuvia värejä. ColorMap-olio antaa värejä LinePlotin käyttöön get_next_color-metodin avulla.

LinePlotDrawer-luokka huolehtii kuvaajan piirtämisestä ja kommunikaatiosta back endin ja PyQt:n välillä. LinePlotDrawer käyttää piirtämiseen apuluokkia:

- VerticalLabel: Pystysuoraan kirjoitettu teksti y-akselin nimeä varten (luokan toteutus otettu Stackoverflow-sivustolta, linkki viitteissä)
- Line: Piirtää suoran viivan kahden pisteen välille. Käytetään käyrien ja gridin piirtämiseen.
- PlotBackground: Valkoinen alue kuvaajan pohjaksi.

Lisäksi luotiin esimerkkisovellusta varten luokat Controls ja RandomWalkDemo. Controls-luokka piirtää kaksi pudotusvalikkoa, jonka avulla voi päättää haluamansa kuvaajan, sekä painikkeen, jolla kuvaajan voi päivittää. RandomWalkDemo on esimerkkisovelluksen pääluokka, joka luo käyttäjän syötteen perusteella LinePlot-olion ja piirtää vastaavan kuvaajan LinePlotDrawerin avulla.

LinePlotDrawer, sen apuluokat, Controls ja RandomWalkDemo perivät kaikki PyQt:n QWidgetin.

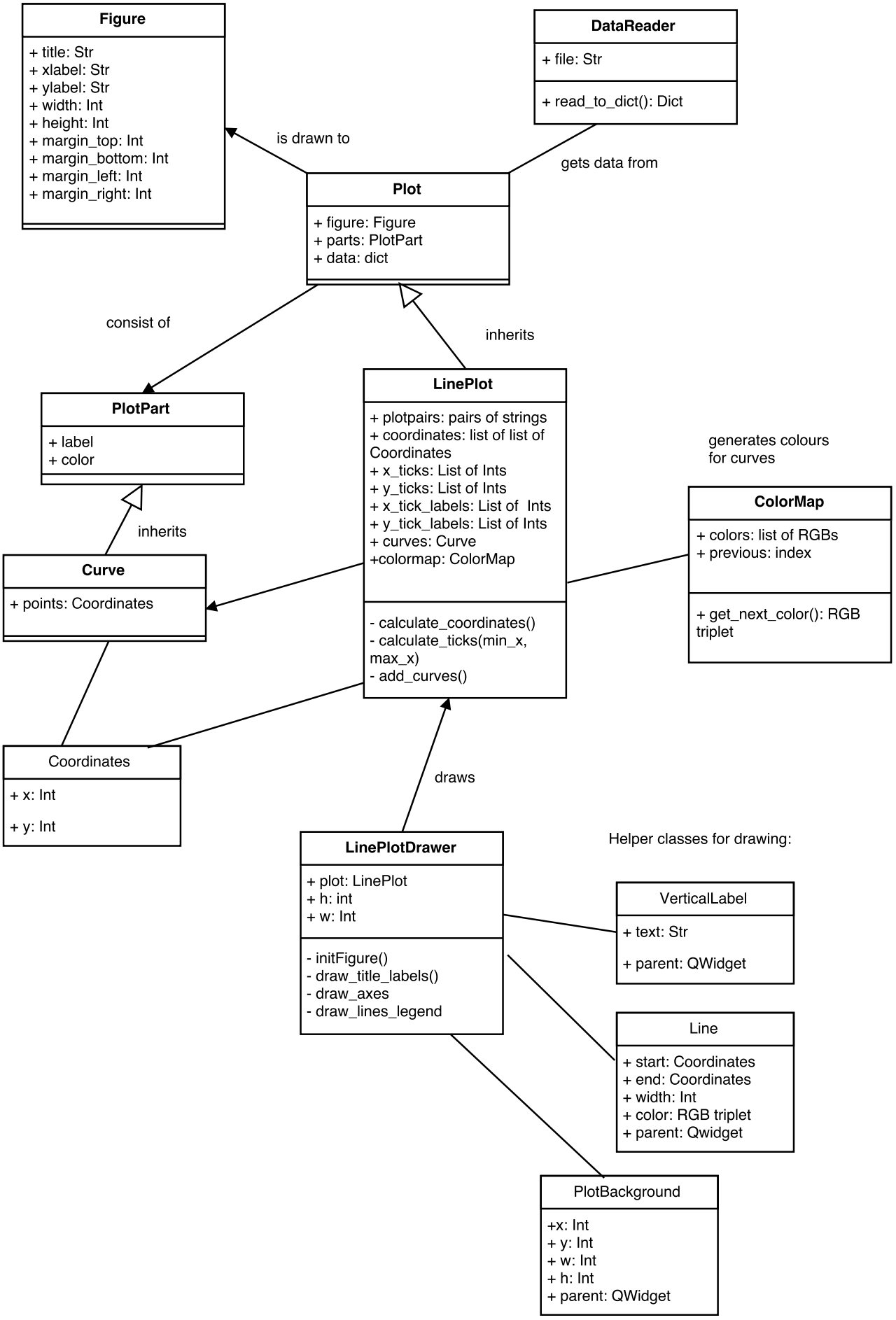


Figure
+ title: Str
+ xlabel: Str
+ ylabel: Str
+ width: Int
+ height: Int
+ margin_top: Int
+ margin_bottom: Int
+ margin_left: Int
+ margin_right: Int
DataReader
+ file: Str
+ read_to_dict(): Dict
is drawn to
gets data from
Plot
+ figure: Figure
+ parts: PlotPart
+ data: dict
consist of
inherits
PlotPart
+ label
+ color
LinePlot
+ plotpairs: pairs of strings
+ coordinates: list of list of Coordinates
+ x_ticks: List of Ints
+ y_ticks: List of Ints
+ x_tick_labels: List of Ints
+ y_tick_labels: List of Ints
+ curves: Curve
+colormap: ColorMap
- calculate_coordinates()
- calculate_ticks(min_x, max_x)
- add_curves()
generates colours for curves
ColorMap
+ colors: list of RGBs
+ previous: index
+ get_next_color(): RGB triplet
inherits
Curve
+ points: Coordinates
Coordinates
+ x: Int
+ y: Int
draws
LinePlotDrawer
+ plot: LinePlot
+ h: int
+ w: Int
- initFigure()
- draw_title_labels()
- draw_axes
- draw_lines_legend
Helper classes for drawing:
VerticalLabel
+ text: Str
+ parent: QWidget
Line
+ start: Coordinates
+ end: Coordinates
+ width: Int
+ color: RGB triplet
+ parent: Qwidget
PlotBackground
+x: Int
+ y: Int
+ w: Int
+ h: Int
+ parent: QWidget

Algoritmit

Kaikissa algoritmeissa oletetaan, että kuvaajan korkeus ja leveys ovat yli 20 pikseliä.

Ohjelma muuntaa datapisteiden arvot Coordinates-olioiksi eli pikselikoordinaatistoon. X-akselin arvoille kaava on muotoa:

$$X_{pixels} = (X_{raw} - min(X_{raw})) * \frac{Figure.width - 20}{max(X_{raw}) - min(X_{raw})} + Figure.margin\_left + 10$$

Muunnos skaalaa x-akselin arvot siten, että minimi sijaitsee 10 pikseliä itse kuvaajan vasemmasta reunasta oikealle ja maksimi 10 pikseliä kuvaajan oikeasta reunasta vasemmalle. Tällöin käyrät eivä kosketa kuvaajan oikeaa reunaa tai y-akselia.

Y-akselin arvoille vastaava muunnos on:

$$Y_{pixels} = (Y_{raw} - max(Y_{raw})) * \frac{Figure.height - 20}{min(Y_{raw}) - max(Y_{raw})} + Figure.margin\_top + 10$$

Muunnos skalaa Y-akselin arvot siten, että maksimi sijaitsee 10 pikseliä kuvaajan yläreunasta alaspäin ja minimi 10 pikseliä kuvaajan alareunasta ylöspäin.

Jos kuvaajaan piirretään useampi käyrä, minimi- ja maksimiarvoina käytetään kaikkien käyrien globaaleja minimi- ja maksimiarvoja.

Vastaavia minimi- ja maksimiarvoja käytetään myös, kun etsitään sopivat arvot merkattavaksi akseleille. Algoritmin pseudokoodi:

```
approx_number_of_ticks = 5
interval = (max(data) - min(data)) / approx_number_of_ticks
correct_power_of_10 = (floor(log10(interval)))^10

if correct_power_of_10 = 0
    step = round(interval)
else if correct_power_of_10 < 0
    step = correct_power_of_10 * round(interval / correct_power_of_10)
else
    step = correct_power_of_10 * round(interval / correct_power_of_10)

start = step * floor(min(data) / step)
end = step * floor(max(data) / step)
ticks = values from start to end by step

if first tick is outside of plot area (too small)
    pop first tick
if (last tick + step) can fit to plot area
    add (last tick + step) to ticks
```

Viimeiset tarkistukset ovat tarpeellisia, sillä datapisteiden viemän tilan lisäksi kuvaajassa on 10 pikselin kokoiset tyhjät alueet joka puolella pisteitä. Algoritmi varmistaa, että tähän tilaan voidaan piirtää merkintä akselille ja vastaava gridin viiva, jos arvot sen sallivat. Jos akselilta puuttuisi siihen kuuluvia arvoja, kuvaaja ei olisi enää yhtä luotettava.

Tietorakenteet

Tiedostosta luettu data voidaan tallettaa monessa eri muodossa. Tämä ohjelma käyttää Pythonin sanakirjaa (dict), jossa jokainen avain on muuttujan nimi ja vastaava arvo lista, joka sisältää datapisteiden arvot. Sanakirjassa muuttujien nimet kulkevat kätevästi mukana, eikä niitä tarvitse tallettaa erikseen. Ohjelmoinnin yksinkertaistamiseksi muut tiedot tallennettiin listoina. Ohjelmassa ei ole itse ohjelmoituja tietorakenteita.

Tiedostot

Ohjelma lukee datan .csv-tiedostosta. Tiedoston alussa on otsikkorivi, joka sisältää muuttujien nimet. Jokainen sarake vastaa yhtä muuttujaa, ja jokaisessa sarekkeessa tulee olla saman verran arvoja. Muuttujien arvot voiva olla joko tekstiä tai numeerisia. Ohjelma ei käsittele puuttuvia arvoja.

Testaus

Ohjelmaa ei aikataulusyistä ole testattu täysin kattavasti. Yksikkötestejä on tehty DataReader-luokalle sekä osalle LinePlot-luokan metodeista. Muut osat ohjelmasta on testattu lähinnä manuaalisesti piirtämällä dataa muutamasta eri tiedostosta.

Testaukseen olisi pitänyt varata enemmän aikaa. Tämän epäonnistuminen johtui lähinnä ajanpuutteesta eikä niinkään suunnitteluvirheestä.

Tunnetut puutteet ja viat

Ohjelman suurimpia puutteita on akseleille merkittyjen arvojen ja selitteessä esiintyvien nimien sijoittelu kuvaajaan. Tällä hetkellä niille varattu tila on määritetty kiinteäksi määräksi pikseleitä. Tämä ei ole kovin joustava ratkaisu, joten varsinkin hyvin pitkät arvot y-akselilla tai pitkät muuttujien nimet selitteessä voivat aiheuttaa hankaluuksia. Tämän ongelman voisi korjata käyttämälä paremmin hyödyksi PyQt:n tarjoamia joustavampia asettelutapoja.

Lisäksi perusteellisen testauksen puutteen vuoksi ohjelmassa saattaa esiityä bugeja, varsinkin jos ohjelmalle annetaan esimerkiksi tekstiä lukujen sijaan tai arvot ovat muuten epätavallisia. Riittävä ääritapuasten testaus tekisi ohjelmasta varmemmin toimivan.

Parhaat ja heikoimmat kohdat

Yksi ohjelman parhaista puolista ovat sen käyttämät algoritmit. Koordinaatistomuunnokset ovat yksinkertaisia lineaarisia operaatioita, mutta hyvien arvojen valitseminen akseleille ei ollut aluksi yksinkertaista. Arvot valitseva algoritmi toimii hyvin itseisarvoltaan pienillä ja suurilla luvuilla.

Ohjelman tuottamat kuvaajat ovat visuaalisesti onnistuneita. Valmiiden värikarttojen käyttö varmistaa värien olevan hyvin eroteltavissa ja gridi erottuu juuri sopivasti.

Ohjelman heikoin puoli on edellisessä osiossakin mainittu tekstin sijoittelu kuvaajan ulkopuolelle. Kaiken kuvaajan ulkopuolella olevan asettelua olisi varaa parantaa, jotta ohjelma olisi luotettavampi.

Toinen merkittävä heikkous on se, ettei kuva skaalaudu, vaan piirtyy vakiokokoisena pikseleissä. Skaalautuvan kuvaajan toteutus olisi haastavaa, sillä datapisteiden paikat tulis laskea uudelleen aina kuvaajan koon muututtua.

Poikkeamat suunnitelmasta

Alkuperäiseen suunnitelmaan verrattuna visualisointikirjastossa on huomattavasti vähemmän luokkia. Assistentin neuvon mukaan kommunikaatio back endin ja PyQt:n välillä tapahtuu vain yhden luokan kautta, eikä jokaisesta luokasta, kuten alkuperäisessä suunnitelmassa. Tämän takia useita suunnitelman luokkia ei ollut tarpeen toteuttaa, sillä ne olisivat lähinnä johtaneet saman tiedon päällekkäiseen tallennukseen.

Sunnitelmasta poiketen visualisointikirjaston back end kirjoitettiin ensin lähes valmiiksi ja PyQt-osuus tehtiin vasta sen jälkeen. Varsinkin PyQt:n käytön laajempaan opetteluun kului melko paljon aikaa.

Toteutunut työjärjestys

1.4. - 13.4.	Datan lukeminen tiedostosta, back endin luokkien hahmottelua
14.4 - 19.4.	Suurin osa back endistä valmis
20.4. - 25.4.	PyQt-osuus, kuvaajan piirtäminen
25.4 - 27.4.	Back endin muutokset ja yhteensovitus kuvaajan piirron kanssa
27.4. - 28.4.	Kuvaajan liittäminen osaksi esimerkkisovelluksen käyttöliittymää
29.4. - 1.5.	Wapputauko
2.5. - 4.5.	Käyttöliittymän viimeistely, koodin siistiminen ja dokumentointi
5.5.	Dokumentin viimeistely

Arvio lopputuloksesta

Visualisointikirjasto tarjoaa kaiken toiminnallisuuden, jota siltä edellytettiin. Suurimmaksi puutteeksi jää kuvaajan ulkopuolella olevan tekstin huono asettelu. Jos projektia jatkettaisiin, tämä olisi ensimmäinen korjattava asia. Itse kuvaajat ovat kuitenkin toimivia ja visuaalisesti hyvälaatuisia.

Käytettävyyden kannalta suurin ongelma on, että kuvaajaan piirrettävien muuttujien nimien kirjoittaminen kaksiulotteiseen listaan on melko työlästä. Tätä voisi myös kehittää.

Koodin tasolla parantamisen varaa on aina. Varsinkin LinePlot luokka paisui melko suureksi ja sen useat kentät voivat tehdä toteutuksesta hieman sekavan.

Ohjelman rakenne mahdollistaa uusien kuvaajatyyppien lisäämisen. Figure-luokka sekä PyQt-puolen apufunktiot helpottavat uusien kuuvaajatyyppien lisäämistä ja DataReader pystyy lukemaan myös tekstimuotoista dataa.

Viitteet

Pythonin dokumentaatio: `https://docs.python.org/3.5/library/`

Qt:n dokumentaatio: `http://doc.qt.io/qt-5/`

ZetCoden PyQt5-tutoriali: `http://zetcode.com/gui/pyqt5/`

VerticalLabel-luokan toteutus: `http://stackoverflow.com/questions/3757246/pyqt-rotate-a-qlabel-so-that-its-positioned-diagonally-instead-of-horizontally`

Värikartat ja Python-rajapinnan dokumentaatio: `http://colorbrewer2.org/` `https://pypi.python.org/pypi/brewer2mpl/1.4`

numpy-paketin dokumentaatio: `https://docs.scipy.org/doc/numpy/reference/`