VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Lauri Mäntylä

DATAN VISUALISOINTI PYTHONILLA

Liiketalous

2021

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittely

TIIVISTELMÄ

Tekijä	Lauri Mäntylä
Opinnäytetyön nimi	Datan visualisointi Pythonilla
Vuosi	2021
Kieli	suomi
Sivumäärä	32
Ohjaaja	Klaus Salonen

Tämän opinnäytetyön tavoitteena oli selvittää, miten datan visualisointisovellus luodaan Python-ohjelmointikielellä. Tavoitteena oli luoda verkkosovellus, jossa visualisoidaan dataa amerikkalaisen jääkiekkosarjan NHL:n joukkueista ja pelaajista. Lisäksi tutkitaan mitä visualisointi on ja mitä visualisoinnin prosessiin kuuluu.

Toisessa luvussa käydään läpi visualisoinnin määritelmää, visualisointiprosessia sekä sen hyötyjä ja haittoja. Kolmannessa luvussa kerrotaan Dash-ohjelmistokehyksestä, sen historiasta ja toiminnasta. Neljännessä luvussa luodaan visualisointisovellus. Viidennessä luvussa on tehty päätelmät sekä pohditaan sovelluksen jatkokehitysmahdollisuuksia. Työssä on käytetty lähteenä Plotlyn ohjesivua, verkkoartikkeleita sekä visualisointia käsittelevää kirjallisuutta.

Opinnäytetyön lopputuloksena oli toimiva verkkosovellus, joka visualisoi urheiludataa erilaisiin kaavioihin. Työn valmistuttua saatiin vastaus työlle asetetuille kysymyksille.

Avainsanat Python, data, visualisointi, Dash, ohjelmointi

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietojenkäsittely

ABSTRACT

Author	Lauri Mäntylä
Title	Data visualization with Python
Year	2021
Language	Finnish
Pages	32
Name of Supervisor	Klaus Salonen

The goal of this thesis was to find out how to create a data visualization program in Python programming language. The objective was to create a web app, which visualizes data from NHL teams and players. In addition, this thesis studied what visualization is and what is involved in the visualization process.

In the second chapter data visualization, the visualization process and the pros and cons of visualization are discussed. The third chapter examines the Dash framework, its history, and operating principles. The creation of the data visualization app is described in the fourth chapter. The fifth chapter draws conclusions and discusses possibilities for further development. This thesis uses Plotly's website, online articles, and visualization literature as its sources.

The outcome of this thesis was a working web app, which visualizes sports data into different charts. Upon completion of the thesis, the questions asked in the thesis were answered.

Keywords Python, data, visualization, Dash, programming

SISÄLLYS

KUVIO- JA TAULUKKOLUETTELO

1 JOHDANTO

Tämän opinnäytetyön tavoitteena on luoda verkkosovellus, joka muuttaa tilastodataa visuaaliseen muotoon. Tähän käytetään Python-ohjelmointikieltä ja Dash-ohjelmistokehystä. Samalla tutkitaan miten nämä työkalut soveltuvat visualisointisovelluksen luomiseen. Käydään myös läpi mitä visualisointi on ja mitä asioita kuuluu visualisoinnin prosessiin.

Idea tähän aiheeseen syntyi urheilusta, jossa erilaisia tilastoja kerätään paljon. Niitä on mielenkiintoista tutkia ja vertailla, joten aion myös tässä työssä käyttää tilastoja urheilusta. Lisäksi halusin opetella käyttämään Pythonia.

Dash on Python, R ja Julia kielille rakennettu avoimen lähdekoodin ohjelmistokehys, jota käytetään analyyttisten verkkosovelluksien kehittämiseen. Se on yksi suosituimmista visualisointikirjastoista.

Opinnäytetyön on tarkoitus vastata seuraaviin kysymyksiin; mitä on datan visualisointi, miten Python-ohjelmointikielellä luodaan visualisointisovellus ja miten Python soveltuu datan visualisointiin?

2 VISUALISOINTI

Visualisointeja näkee päivittäin monessa eri muodossa esimerkiksi kaaviona sanomalehdessä, metrokarttana sekä säätiedotuksissa. Näissä tapauksissa visualisointia käytetään vaihtoehtona tai täydennyksenä kirjalliselle tai verbaaliselle tiedolle. Datan määrän kasvaessa on yhä suurempi tarve sen visualisoinnille ja yhä suurempi kysyntä visualisointityökaluille ja -tekniikoille.

2.1 Mitä visualisointi on

Termiä ”visualisointi” käytetään usein sekä kuvaamaan prosessia, josta tieto muutetaan visuaaliseen muotoon, että visualisointiprosessin lopputulosta (Koponen, Hilden & Vapaasalo 2016, 23.). Visualisointitutkijan Robert Kosaran (2007, 2) mukaan visualisoinnille ei kuitenkaan ole yhtä yleisesti hyväksyttävää termiä. Kosara onkin antanut termille seuraavanlaiset kriteerit:

- Visualisointi perustuu (ei-visuaaliseen) dataan.
- Visualisointi tuottaa kuvan.
- Tulos on luettava ja tunnistettava.

Visualisoinnilla tarkoitetaan siis tilastojen, normaalisti näköaistin tavoittamattomissa olevan tiedon kuten esimerkiksi sisäelinten rakenteen tai muun abstraktin tiedon visuaalista esittämistä. Koponen, Hilden ja Vapaasalo jakavat tietoa välittävät kuvat kahteen pääkategoriaan riippuen niiden viestinnällisestä funktiosta, visualisointeihin ja infografiikkoihin. Visualisoinnin tavoitteena on abstraktin tiedon esittäminen, jonka pohjalta lukija voi tehdä johtopäätöksiä. Visualisointi on eksploratiivista, uusia piirteitä paljastavaa grafiikkaa. Se ei pyri esittämään tekijän ennalta määritettyä viestiä, vaan toimia välineenä, jonka avulla lukija voi itse selvittää mielenkiintoisia piirteitä. Infografiikka on selittävää grafiikkaa eli sen ensisijainen tehtävä on kertoa lukijalle tietoa. Toisin kuin visualisoinnit, infografiikka luodaan yleensä käsin jonkun ennalta määritetyn viestin välittämistä varten. Ko-

ponen, Hilden ja Vapaasalo muotoilevat eron kategorioiden välillä näin: ”Infografiikka kertoo tarinan, kun taas visualisointi on väline, jonka avulla lukija voi löytää oman tarinansa aineiston uumenista.” Jako näihin kategorioihin ei kuitenkaan ole mustavalkoinen. Niitä voidaan ajatella jatkumona tarkkarajaisten kategorioiden sijaan. Grafiikat sisältävät usein sekä infografiikan että visualisoinnin piirteitä. (Koponen ym. 2016, 20–23.)

Kaikki visuaalinen viestintä ei kuitenkaan pyri välittämään tietoa. Viestinnän tavoite voi liittyä myös arvoihin, mielikuvaan tai muuhun vähemmän konkreettiseen asiaan. Tällöin visualisoinnit ja infografiikat ovat vääriä välineitä siihen tarkoitukseen. (Koponen ym. 2016, 23.)

Tietosuunnittelija Albert Cairon mukaan infografiikan tavoitteena ei ole tehdä luvuista mielenkiintoisia vaan muuttaa ne visuaalisiin muotoihin, joista ihmisaivot voivat löytää merkityksiä. Infografiikkaa ei ole tehty nähtäväksi ja passiivisesti omaksuttavasti, vaan se on visuaalinen laite, jota tulisi pystyä käsittelemään, lukemaan ja analysoimaan tarinan paremmin ymmärtämiseksi. Infografiikka ei ole pelkästään kaunis kuva, vaan visuaalinen esitys todisteista. (Kuenn 2013)

2.2 Visualisoinnin prosessi

Datan visualisointi jaetaan neljään vaiheeseen, johon kuuluu muutamia palautesilmukoita:

1. Datan kerääminen
2. Esikäsittely, jossa data muutetaan helposti käsiteltäväksi ja yleensä myös karsitaan dataa, niin että saadaan haluttuja piirteitä näkyviin datasta.
3. Valitun datan kartoitus visuaaliseen muotoon. Tämä tehdään tietokone algoritmeilla, jotka luovat ruudulle kuvan. Käyttäjän syöte voi muuttaa kartoituksia, korostaa tiettyjä osajoukkoja tai muuttaa esityksen ulkoasua.
4. Ihmisen havainnointi- ja kognitiivinen järjestelmä. (Ware 2012)

Palautesilmukoista pisin liittyy tiedon keräämiseen. Lukija saattaa ryhtyä keräämään lisää tietoa aiheesta visualisoinnista saadun tiedon perusteella. Toinen silmukka koskee tiedon esikäsittelyä. Visualisoinnin laatijasta saattaa tuntua siltä, että data menettää tarkoituksensa, jos sitä käsitellään tietyllä tavalla. Joskus tähän liittyy haravointia suuren datamäärän lävitse etsien yhtä tärkeää tietoa. Visualisointi voi myös olla hyvin interaktiivinen. Kolmiulotteisissa visualisoinneissa käyttäjä voi ”lentää” sen lävitse etsien eri näkökulmia. Visualisoinneissa voi olla myös eri parametreja, joista käyttäjä voi valita mielenkiintoisimmat. (Ware 2012)

2.3 Visualisoinnin hyödyt

Ihmiset ovat visuaalisia olentoja ja meidän aivomme ovat kehittyneet tuhansien vuosien aikana käsittelemään visuaalista tietoa kauan ennen kirjoitetun tekstin yleistymistä. Siksi ihmisen aivot prosessoivat visuaalista tietoa 60 000 kertaa nopeampaa kuin tekstiä ja jopa 90 % aivoille välitetystä tiedosta on visuaalista. Kuvat ovat kuin visuaalisia oikoreittejä aivoille. (Gioglio & Walter 2014)

Kallur Rahmanin (2021) mukaan visualisoinnin hyödyt ovat seuraavat:

- Visuaalinen data helpottaa nopeaa päätöstentekoa. Tätä varten datan on oltava nopeasti ja yksinkertaisesti ymmärrettävissä, missä datan visualisointi auttaa.
- Visualisointi auttaa yhteenvedon jakamisen kanssa. Hyvän visualisoinnin avulla voidaan välttää pitkien tekstien lukemista ja sen sijaan päätellä kaikki se tieto yhdestä visuaalisesta yhteenvedosta.
- Liiketoiminnassa visualisointien hyödyntämisestä on paljon apua. Sen avulla voidaan mitata tärkeimpiä lukuja sekä mittareita ja näin saada parempia näkökulmia näihin.
- Visualisointi auttaa tunnistamaan kuviot, raot ja trendit. Näiden tunnistamisen avulla voidaan selvittää syitä ja keinoja ja näin tehdä parempia päätöksiä, oli kyse sitten liiketoiminnasta tai muunlaisesta toiminnasta.

- Visualisointi auttaa tarinan kertomisessa. Hyvällä visualisoinnilla voidaan tuoda esiin kuvioita ja trendejä datasta, mikä auttaa halutun tarinan kertomisessa.
- Visualisointia voidaan hyödyntää brändin rakentamisessa. Tarinojen kertominen näyttävien visualisointien avulla voi vahvistaa brändiä. Esimerkiksi Coca-Cola yhdistetään punaiseen väriin näiden vuoksi.
- Visualisoinnin avulla voidaan kertoa paljon vähällä. Yksinkertaisella visualisoinnilla voidaan viestiä paljon enemmän kuin tekstillä tai pelkällä datalla. Tarina voidaan kertoa yhdellä visualisoinnilla.

Colin Waren (2012) mukaan hyödyt ovat seuraavat:

- Visualisoinnin suurin hyöty on se, kuinka paljon tietoa voidaan saada kuvasta irti lyhyessä ajassa. Sitä hyödyntäen voidaan miljoonistakin eri mittauksista saada tärkeät tiedot esille nopeasti.
- Visualisointi helpottaa sellaisten asioiden löytämistä datasta mitä ei osattu odottaa.
- Visualisoinnin ongelmat paljastavat itsensä nopeasti. Virhe datassa tai visualisoinnissa erottuu lukijalle heti muusta datasta. Tästä syystä visualisoinnit voivat olla tärkeitä laadunhallinnassa.
- Visualisointi helpottaa suurten ja pienten piirteiden ymmärtämistä datasta. Erityisesti piirteiden yhdistävien kuvioiden havaitsemiseen.
- Visualisointi helpottaa hypoteesin muodostamista.

2.4 Visualisoinnin haitat

Visualisoinnissa on myös haittapuolensa. Visualisoinnin luo ihminen, minkä takia niiden luomisessa tulee joskus virheitä. Lukija voi hämmentyä tai ymmärtää visualisoinnin väärin, jos se on suunniteltu huonosti. Visualisoinnin luomiseen valittu data voi olla hyvin puolueellista, jolloin lukijaan voidaan vaikuttaa. Visualisoinnit eivät voi auttaa lukijaansa, mikä johtaa siihen, että eri ihmiset voivat ymmärtää

saman kuvan eri tavalla. Samoin jos lukija on keskittynyt väärään asiaan visualisoinnissa, hän ei ehkä ymmärrä sen kertomaa tarinaa. (Hoffman 2021)

3 DASH

Dashin kehittäjä on kanadalainen Plotly. Yritys perustettiin vuonna 2013 ja jo vuonna 2015 Plotlyn Python ja R kirjastot olivat maailman ladatuimmat graafiset kirjastot. Samana vuonna Dashin prototyyppi julkistettiin GitHubissa. Sitä esiteltiin pankeille, laboratorioille sekä datatieteen ammattilaisille ja tuotetta kehitettiin heiltä saadun palautteen mukaan. Vuonna 2017 Dashin betaversio lanseerattiin ja kehys julkaistiin vuonna 2019. (Plotly 2021a, 2021b)

Dash on Python, R ja Julia ohjelmointikielille rakennettu avoimen lähdekoodin ohjelmistokehys, jolla voidaan rakentaa verkkoanalyysisovelluksia. Sen avulla voidaan tehdä datan analysointia, tutkimista, visualisaatioita, mallintamista, instrumenttien hallintaa ja raportointia. Dashin avulla luodut sovellukset toimivat verkkoselaimella, jonka ansiosta luotuja sovelluksia on helppo jakaa sekä ne myös toimivat usealla eri laitteella. (Plotly 2017)

Dash on rakennettu Flask, Plotly.js ja React.js kirjastojen päälle. Se käyttää Reactia luodakseen verkkosivulle käyttöliittymän. Tällä käyttöliittymällä voidaan hallita ja tutkia Plotlyllä luotuja kaavioita ja visualisointeja. Dash tukee myös muita ulkoisia visualisointikirjastoja kuten seabornia ja altairia, mutta näillä luodut kaaviot eivät ole yhtä interaktiivisia kuin Plotlyllä luodut kaaviot. (Kilcommins 2021)

Dashin toiminta perustuu kolmeen perusosaan, Dash komponentteihin, Plotlyn kaavioihin ja callbackiin eli vastakutsuun. Dash komponentteja ovat napit, liukusäätimet, pudotusvalikot yms. Näiden avulla visualisoinneista luodaan interaktiivisia. Plotlyn avulla luodaan sovellukseen visualisoinnit, nämä voivat olla esim. kaavioita tai karttoja. Vastakutsu luo yhteyden komponenttien ja kaavioiden välille, mahdollistaen näiden yhteistyön.

Luotuja kaavioita voidaan käsitellä ja rajata eri tavoin. Kaavion ylälaidasta löytyy työkaluja, jotka mahdollistavat tämän. Kaaviota voidaan zoomata, siirrellä sekä datapisteitä voidaan valita laatikko- tai lassotyökalulla. Käsitelty kaavio voidaan tallentaa PNG-kuvana työkalurivistä löytyvällä latauspainikkeella.

Dashistä on myös olemassa yrityskäyttöön suunnattu maksullinen versio Dash Enterprise. Tämä versio tarjoaa työkaluja, jotka helpottavat käyttöönottoa ja suurien tietomäärien kanssa työskentelyä. Enterprisen mukana tulevalla Design Kitillä voi voi suunnitella ja toteuttaa Dash sovellusten käyttöliittymiä. (Plotly 2017c)

4 SOVELLUKSEN LUOMINEN

Käytännön osuuden tavoitteena on luoda verkkosovellus, joka visualisoi NHL-pelaajien ja joukkueiden tilastodataa. Visualisoinnin ensimmäisessä vaiheessa kerätään dataa. Data haetaan Natural StatTrick sekä QuantHockey sivuilta. QuantHockey sivulta haetaan parhaiden pistemiesten tilastot ja Natural StatTrickistä loput. Näiltä sivuilta tilastot ovat helposti ladattavissa CSV-muodossa.

Työ aloitetaan tuomalla tarvittavat kehykset ja kirjastot sovellukseen. Nämä ovat Pandas, Plotly Express, Pillow sekä Dash ja sen komponentit: core components, html components ja bootstrap components. Pandasin avulla luetaan dataa sekä syötetään niitä dataframeihin eli datakehyksiin. Plotly Express on korkean tason rajapinta, jolla luodaan kaaviot ja kartat. Pillow-kirjastosta käytetään Image-moduulia kuvien lisäämiseksi kaavioihin. Dash core components sisältää Dashin komponentit, jotka mahdollistavat kaavioiden interaktiivisuuden. HTML ja Bootstrap components mahdollistavat nimiensä mukaisesti HTML:än sekä Bootstrapin käytön sovelluksessa.

4.1 Datan lukeminen

Visualisoinnin toisessa vaiheessa käsitellään dataa luettavaan muotoon sekä karsitaan mahdolliset turhat asiat pois. Ladatut tiedot ovat CSV-muodossa, eli pilkulla tai jollain muulla välimerkillä erottua, riveittäin tallennettua tekstidataa. Data luetaan Pandasin read_csv funktiolla ja syötetään datakehyksiin. Datakehys on kaksiulotteinen taulukkorakenne, jossa on merkityt rivit ja sarakkeet (Pandas 2021). Datakehyksiä on yksi jokaista CSV-tiedostoa kohti. Nämä tiedostot on jaettu tilastojen mukaan ja niistä käytetään aina sitä mitä tilastoja halutaan tutkia. Nrows-parametrillä määritellään, kuinka monta riviä halutaan lukea ja sep-parametrillä määritellään mitä välimerkkiä CSV-tiedostossa käytetään datan erotteluun.

```
dfPoints = pd.read_csv("Points.csv", nrows=50, sep=";")
dfGoals = pd.read_csv("Goals.csv", nrows=50, sep=";")
```

Kuva 1. Datan Lukeminen.

4.2 Rakenne

Sovellukseen luodaan Dashin Bootstrap komponentteja käyttäen navigointipalkki sivun ylälaitaan, johon lisätään linkit eri kaavioihin. Palkkiin luodaan kaksi alasvetovalikkoa, ”Pelaajat” ja ”Joukkueet”. Luodaan muuttuja joka sisältää palkin ja sijoitetaan se app.layout-muuttujaan. Tänne sijoitettu sisältö tulee sovellukseen näkyväksi.

```
navbar = dbc.NavbarSimple(
    children=[
        dbc.NavItem(dbc.NavLink("Koti", href="/")),
        dbc.DropdownMenu(
            children=[
                dbc.DropdownMenuItem("Pelaajat", header=True),
                dbc.DropdownMenuItem("Pisteet", href="/pisteet"),
                dbc.DropdownMenuItem("Maalit", href="/maalit"),
            ],
            nav=True,
            in_navbar=True,
            label="Pelaajat",
        ),
        dbc.DropdownMenu(
            children=[
                dbc.DropdownMenuItem("Joukkueet", header=True),
                dbc.DropdownMenuItem("Joukkuetilastot", href="/joukkue"),
                dbc.DropdownMenuItem("Kotimaat", href="/joukkue2"),
            ],
            nav=True,
            in_navbar=True,
            label="Joukkueet",
        ),
    ],
    brand="Tilastot",
    brand_href="#",
    color="primary",
    dark=True,
)
```

Kuva 2. Navigointipalkki.

Valikot saadaan toimimaan vastakutsun avulla. Vastakutsuun määritellään yksi tai useampi input ja output. Input liitetään johonkin sovelluksen osaan, jonka muuttuessa vastakutsu kutsutaan ja se antaa tiedon outputissa määriteltyyn sijaintiin. Palautettu tieto määritellään funktiossa, joka määritellään vastakutsun jälkeen. Tässä tapauksessa (Kuva 3) inputtiin määritellään url ja pathname ja outputtiin

page-content ja children. Tämä tarkoittaa sitä, että kun URL-osoite muuttuu, vastakutsu antaa funktion palauttaman tiedon sovelluksen osalle, jonka ID on page-content ja tieto sijoitetaan sen children-parametriin. (Ploty 2021d)

```
@app.callback(
    Output("page-content", "children"),
    Input("url", "pathname")
)

def render_page_content(pathname):

    if pathname == "/":

        return indexLayout

    elif pathname == "/pisteet":

        return pointsLayout

    elif pathname == "/joukkue":

        return teamsLayout

    elif pathname == "/joukkue2":

        return teamCountryLayout

    # 404
    return dbc.Jumbotron(
        [
            html.H1("404: Not found", className="text-danger"),
        ]
    )
```

Kuva 3. Content-vastakutsu.

Funktiolla render_page_content määritellään tieto, joka annetaan eteenpäin, riippuen siitä, missä käyttäjä on sivulla. Esim. jos käyttäjä on sivulla ”pisteet”, palauttaa funktio pointsLayout-muuttujan sisällön Tämä sijoitetaan content-muuttujaan, mikä taas sijoitetaan app-muuttujan layout-ominaisuuteen (Kuva 4). Sovelluksessa tulee aina näkyviin navigointipalkki ja content, mikä määrittyy osoitteen perusteella. Dcc.Location-komponentti kertoo vastakutsulle, missä osoitteessa käyttäjä on. Mikäli osoitetta ei löydy, annetaan virheilmoitus.

```
content = html.Div(id="page-content", children=[], style=CONTENT_STYLE)

app.layout = html.Div([
    dcc.Location(id="url"),
    navbar,
    content
])
```

Kuva 4. App.layout määrittely

Joka sivulle määritetään oma layout eli sijoittelu. Tänne sijoitetaan kaikki sillä sivulla olevat komponentit ja tekstit sekä luodaan tyhjä paikka kaavioille. Kaaviot luodaan myöhemmin vastakutsuilla. Ensimmäistä kaaviota varten luodaan alasvetovalikko, josta käyttäjä voi valita haluamansa tilaston.

```
pointsLayout = html.Div([
        html.H1('Kaikkien aikojen pistepörssin top 50',
                        style={'textAlign':'center'}),

        dcc.Dropdown(id='pointsDropdown',
                    options=[
                        {"label": "P", "value": "P"},
                        {"label": "G", "value": "G"},
                        {"label": "A", "value": "A"},
                        {"label": "PIM", "value": "PIM"},
                        {"label": "PPG", "value": "PPG"},
                        {"label": "SHG", "value": "SHG"},
                        {"label": "SHOTS", "value": "SHOTS"},],
                    multi=False,
                    value="G",
                    style={'width': "40%"}
                    ),
                dcc.Graph(id='pointsGraph', figure={}),
                dcc.Graph(id='pointsScatter', figure={}),

])
```

Kuva 5. Pistetilastosivun layout.

4.3 Pelaajakaaviot

Visualisoinnin kolmannessa vaiheessa data kartoitetaan visuaaliseen muotoon. Ensimmäisessä kaaviossa vertaillaan eniten pisteitä tehneitä pelaajia. Määritellään kaavio px.bar-funktiolla (Kuva 6). Kaaviosta tulee näin pylväskaavio. Aluksi se määritellään käyttämään dataa dfPoints-datakehyksestä, joka sisältää 50 pelaajan tilastot järjesteltynä tehtyjen pisteiden mukaan. Kaaviosta luodaan interaktiivinen vastakutsun avulla. Y-akseliksi määritellään pelaajan nimi ja x-akseliin sijoitetaan käyttäjän valitsema tilasto. Pylväiden väri määritellään pelaajan pelipaikan mukaan color-parametrilla. Lisäksi määritellään hover-dataksi pelaajan nimi ja tehdyt pisteet, eli käyttäjän liikuttaessa hiiren kaavion pylväiden päälle, tulee valittujen tilastojen lisäksi myös nämä tiedot näkyviin. Lopuksi kaavio järjestetään suurimmasta pienempään update_layout-funktiolla. (Ploty 2021e)

```
@app.callback(
    Output(component_id='pointsGraph', component_property='figure'),
    Input(component_id='pointsDropdown', component_property='value')
)
def update_graph(option_slctd):

    figPoints = px.bar(
    data.dfPoints,
        x= option_slctd,
        y="Name",
        color="Pos",
        height=900,
        hover_data=["P","Name"]
    )
    figPoints.update_layout(yaxis={'categoryorder':'total ascending'})
    return figPoints
```

Kuva 6. Pylväskaavion vastakutsu.

Vastakutsu kuuntelee aikaisemmin luotua alasvetovalikkoa muutosten varalta. Mikäli valittu tilasto muuttuu, palauttaa se update_graph-funktion avulla päivitetyn kaavion layoutissa määriteltyyn paikkaan. Tämän avulla kaaviosta saadaan interaktiivinen.

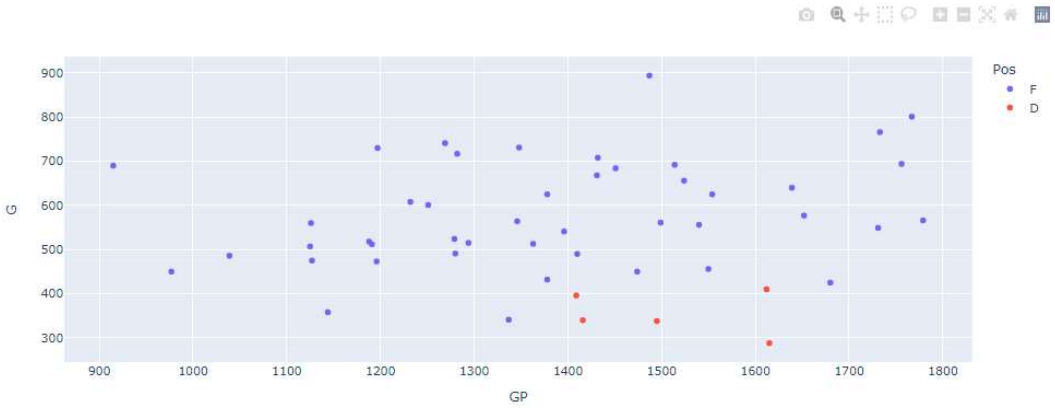
Kuva 7. Pylväskaavio.

Visualisoinnin neljännessä vaiheessa tutkitaan luotua kuvaa ja tehdään päätelmiä siitä. Pylväskaaviosta voidaan päätellä nopeasti parhaimmat pelaajat. Eri tilastoja valitsemalla pystytään vertailemaan pelaajia ja voidaan tehdä johtopäätöksiä heidän pelaamistyylistään. Kuvan 7 visualisoinnista voidaan päätellä esim. se, että listan puolustajat ovat tehneet suurimman osan pisteistään syötöillä. Tämän näkee helposti pylvään koosta ja värityksestä. Saman päätelmän tekeminen kirjoitetusta tiedosta olisi kestänyt huomattavasti kauemmin ja vaatinut enemmän työtä.

Toiseksi kaavioksi valitsin pistekaavion. Sen luominen on samanlaista kuin pylväskaavion. Samaa alasvetovalikkoa hyödynnetään tässä kaaviossa. Kaavio määritellään px.scatter-funktiolla. X-akseliksi valitaan pelattujen otteluiden määrä ja y-akseliksi käyttäjän valitsema tilasto. Vastakutsu palauttaa tiedon samalla tavalla kuin pylväskaaviossa. (Ploty 2021f)

```
@app.callback(
    Output(component_id='pointsScatter', component_property='figure'),
    Input(component_id='pointsDropdown', component_property='value')
)
def update_graph(option_slctd):

    figScatterPoints = px.scatter(
        data.dfPoints,
        x="GP",
        y=option_slctd,
        color="Pos",
        hover_data=["Name"]
        )
    figScatterPoints.update_layout(yaxis={'categoryorder':'total ascending'})
    return figScatterPoints
```

Kuva 8. Pistekaavion vastakutsu.



Kuva 9. Pistekaavio.

Pistekaavio antaa toisenlaisen näkökulman samoihin tilastoihin. X-akseli ei vaihdu ollenkaan, joten eniten otteluita pelanneet pelaajat ovat aina oikealla reunalla. Tämän ansiosta pelaajien vertailu on helpompaa.

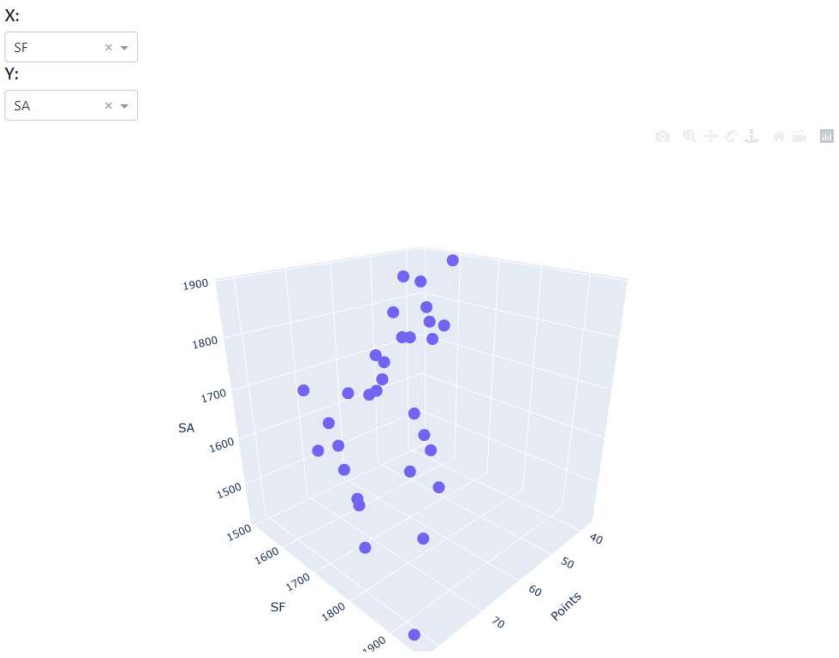
4.4 Joukkuekaaviot

Plotlyn avulla voidaan luoda myös kolmiulotteisia kaavioita Luodaan joukkuetilastoista kolmiulotteinen pistekaavio. Tähän käytetään kauden 2020–2021 tilastoja. Luodaan kaksi alasvetovalikkoa, joihin annetaan arvoiksi joukkuetilastoja, kuten laukaukset, tehdyt maalit, torjuntaprosentti yms. Kaavio luodaan px.scatter_3d-funktiolla sekä yhdistetään valikkoihin vastakutsun avulla (Kuva 10). Koska käyttäjä

voi valita kaavioon kaksi eri arvoa, vastakutsuun lisätään kaksi inputtia. Kun kumman tahansa valikon arvoa muutetaan, niin kaavio päivittyy. X-akseliksi valitaan joukkueen pisteet ja y- sekä z-akseliksi käyttäjän valitsema tilasto. (Ploty 2021g)

```
@app.callback(
    Output(component_id='teamScatter', component_property='figure'),
    Input(component_id='teamDropdownX', component_property='value'),
    Input(component_id='teamDropdownY', component_property='value')
)
def update_graph(option_slctd, option_slctd2):

    figScatterPoints = px.scatter_3d(
        data.dfTeams,
        x="Points",
        y=option_slctd,
        z=option_slctd2,
        height=600,
        hover_data=["Team"]
        )
    return figScatterPoints
```

Kuva 10. Kolmiulotteisen pistekaavion vastakutsu.



Kuva 11. Kolmiulotteinen pistekaavio.

Tällainen kaavio on turhan monimutkainen näiden tilastojen tutkimiseen. Kaksiulotteista pistekaaviota on paljon helpompaa tutkia. Kaaviosta saisi selkeämmän,

mikäli pisteiden tilalle voisi vaihtaa joukkueiden logot. Kolmiulotteiseen pistekaavioon se ei ole mahdollista. Plotlyn avulla voidaan luoda myös kolmiulotteinen viivakaavio.

Luodaan toinen pistekaavio vertaamaan joukkueiden odotettua suorituskykyä ja toteutunutta suorituskykyä (Kuva 12). X-akseliksi valitaan GF% eli goals-for% ja y-akseliksi xGF% eli expected goals-for%. GF% tarkoittaa sitä, kuinka monta prosenttia keskimäärin otteluiden maaleista oli kyseisen joukkueen tekemiä. Jos joukkue voittaa ottelun 3–1, on sen GF% silloin 75 %. xGF% tarkoittaa joukkueen odotettua GF% perustuen joukkueen laukauksiin, maalintekopaikkoihin yms. Kaavion alle lisätään liukusäädin, jolla voidaan valita haluttu kausi. Vastakutsussa if-lauseella tarkastetaan minkä kauden käyttäjä valitsi ja lisätään sen kauden tiedot kaavioon.

Kaavion selkeyttämiseksi luodaan siihen keskiviivat, kuvaustekstit sekä vaihdetaan pisteiden tilalle joukkueiden logot. Kuvaukset ja keskiviivat lisätään add_shape ja add_annotation-metodeilla (Kuva 13). Kuvien lisäämistä varten aluksi poistetaan näkyvistä alkuperäiset pisteet update_traces-metodilla, jossa ne asetetaan läpinäkyviksi (Kuva 14). For-loopissa käydään läpi datakehyksen joukkueiden nimet, korvataan välilyönnit väliviivalla ja lisätään ne muuttujaan team. Add_layout_image-metodilla lisätään kuvat pisteiden tilalle. Ensin kuva haetaan Pillowin Image-moduulin avulla joukkueen nimen perusteella. Xref ja yref asettavat kuvan koordinaattiakselit. Xanchort ja yanchor ankkuroivat kuvan keskelle. Kuvan sijainti määritellään x ja y kohdissa vastaamaan alkuperäisten pisteiden sijaintia, eli GF% ja xGF%. Lopuksi kuvan kooksi valitaan 2. Tämä toistetaan jokaiselle kaavion pisteelle.

```
@app.callback(
    Output(component_id='teamGFScatter', component_property='figure'),
    Input(component_id='slider', component_property='value'),

)
def update_graph(option_slctd):
    dfGF = data.dfTeams20_21
    if option_slctd == 2016:
        dfGF = data.dfTeams16_17
    elif option_slctd == 2017:
        dfGF = data.dfTeams17_18
    elif option_slctd == 2018:
        dfGF = data.dfTeams18_19
    elif option_slctd == 2019:
        dfGF = data.dfTeams19_20
    elif option_slctd == 2020_2021:
        dfGF = data.dfTeams20_21

    figGFScatter = px.scatter(
        dfGF,
        x="GF%",
        y="xGF%",
        height=700,
        hover_data=["Team"]
        )
```

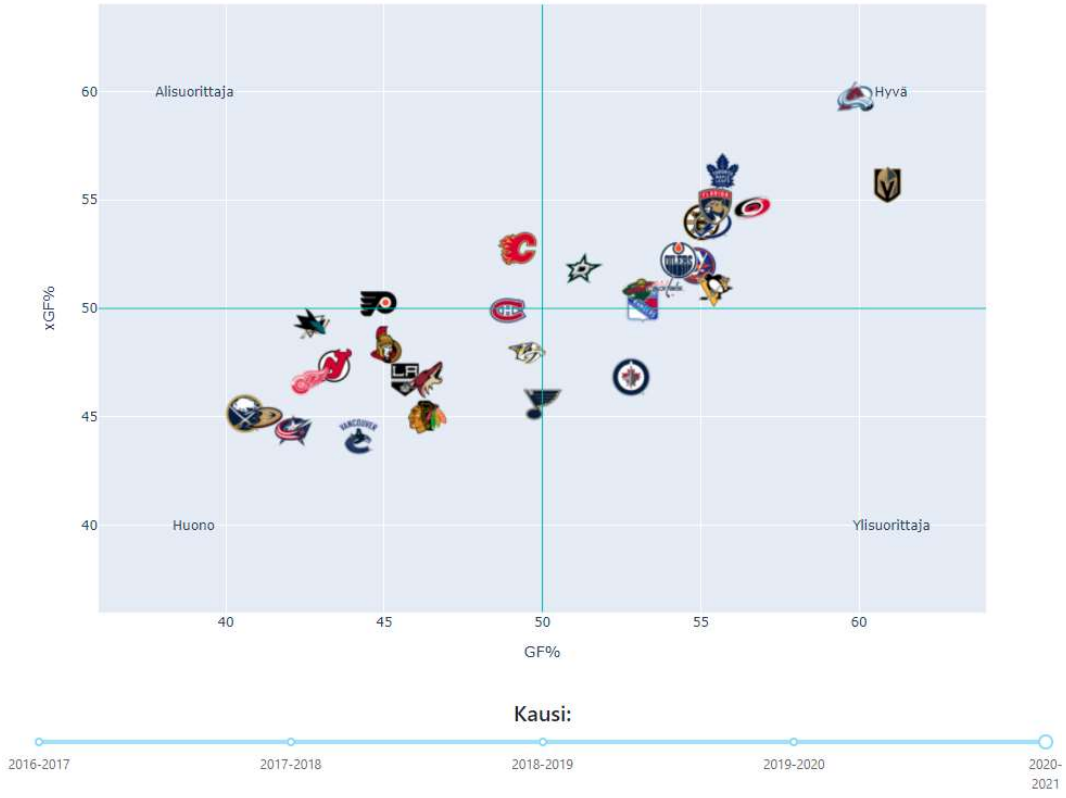
Kuva 12. Joukkuetilastojen pistekaavion vastakutsu.

```
figTeamXGF.add_shape(type="line",
x0=50, y0=38, x1=50, y1=62,
    line=dict(
        color="LightSeaGreen",
        width=1,
    )
)
figTeamXGF.add_shape(type="line",
x0=38, y0=50, x1=62, y1=50,
    line=dict(
        color="LightSeaGreen",
        width=1,
    )
)
figTeamXGF.add_annotation(
        x=39, y=40,
        text="Huono",
        showarrow=False,
)
figTeamXGF.add_annotation(
        x=61, y=60,
        text="Hyvä",
        showarrow=False,
)
```

Kuva 13. Viivojen ja kuvausten lisäys.

```
figGFScatter.update_traces(marker_color="rgba(0,0,0,0)")

for i, row in dfGF.iterrows():
    team = row['Team'].replace(" ", "-")
    figGFScatter.add_layout_image(
        dict(
            source=Image.open(f"images/{team}.png"),
            xref="x",
            yref="y",
            xanchor="center",
            yanchor="middle",
            x=row["GF%"],
            y=row["xGF%"],
            sizex=2,
            sizey=2,
        )
    )
return figGFScatter
```

Kuva 14. Logojen lisääminen.



Kuva 15. Pistekaavio logoilla

Kuvan 15 kaaviosta tulee esille visualisoinnin hyödyt. Logojen lisäämisen ansiosta voidaan nopeasti päätellä, mitkä joukkueet suoriutuivat valitulla kaudella hyvin ja

mitkä eivät. Lisätyt kuvaukset auttavat käyttäjää ymmärtämään, miten joukkueet pärjäsivät.

Dashin avulla voidaan käyttää myös karttoja. Luodaan koropleettikartta, joka näyttää joukkueiden pelaajien syntymämaat. Tätä varten käytettyä dataa tulee hieman käsitellä ennen visualisointia. Ensin luetaan data CSV-tiedostosta, joka sisältää kaikki kaudella 2020–2021 pelanneet pelaajat ja heidän tietonsa sekä sijoitetaan ne datakehykseen (Kuva 16). Usecols-parametrilla rajataan mitä sarakkeita tiedostosta luetaan. Tässä tapauksessa tarvitaan vain joukkueen nimi ja pelaajan syntymämaa, joten käytetään sarakkeita 2 ja 9. Tämän jälkeen groupby-metodilla ryhmitellään pelaajat joukkueittain ja syntymämaittain. Lopuksi vielä lasketaan uuteen sarakkeeseen samasta maasta olleiden yhteislukumäärä size-metodilla ja annetaan sarakkeelle nimi ”count”.

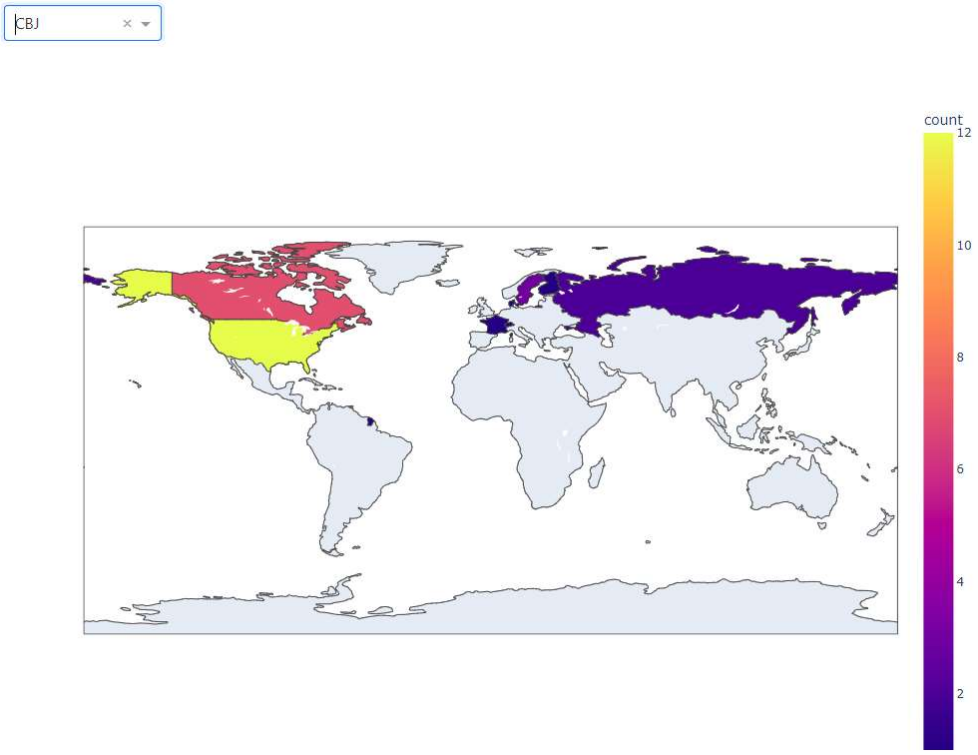
```
dfPlayers = pd.read_csv("Players.csv", sep=",", usecols=[2,9])
dfteamMap = dfPlayers.groupby(['Team', 'Nationality' ]).size().reset_index(name='count')
```

Kuva 16. Datan käsittely.

Luodaan alasvetovalikko johon lisätään joukkueiden nimet. Yhdistetään tämä vastakutsun avulla koropleettikarttaan. Määritellään kartta px.chloropleth-funktiolla (Kuva 17). Määritellään se käyttämään ISO-3 standardin mukaista nimitystä valtioista. Syntymämaa haetaan nationality-sarakkeesta sekä kartan väritys haetaan aikaisemmin luodusta count-sarakkeesta. Kartassa maan väritys muuttuu riippuen siitä, kuinka monta pelaajaa sieltä on kotoisin. (Ploty 2021h)

```python
@app.callback(
    Output(component_id='map', component_property='figure'),
    Input(component_id='team', component_property='value')
)


def update_graph(option_slctd):

    dfteamMap = data.dfteamMap[data.dfteamMap['Team'] == option_slctd]

    figMap = px.choropleth(
    data_frame=dfteamMap,
    locationmode='ISO-3',
    locations='Nationality',
    color='count',
    height=800,

    )
    return figMap
```

Kuva 17. Koropleettikartan vastakutsu.

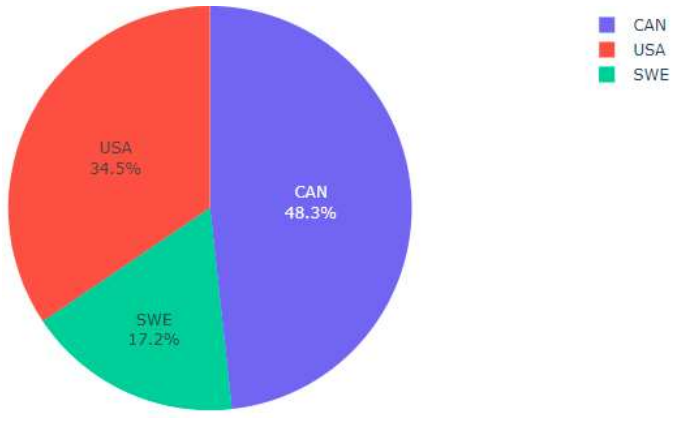


Kuva 18. Koropleettikartta.

Lisätään samasta tilastosta ympyräkaavio. Se luodaan px.pie-funktiolla (Kuva 19). Kaavion arvoina käytetään count-saraketta ja nimet haetaan nationality-sarakkeesta. Kaavio yhdistetään vastakutsun avulla samaan alasvetovalikkoon kuin kartassakin. Vakiona ympyräkaavion sisällä ei ole nimiä, joten lisätään ne update_traces-funktiolla selventämään kaaviota. Lopputuloksena alasvetovalikosta valitaan joukkue, jonka jälkeen sekä kartta että ympyräkaavio päivittyvät näyttämään sen joukkueen tilastoja. (Ploty 2021i)

```
@app.callback(
    Output(component_id='pie', component_property='figure'),
    Input(component_id='team', component_property='value')
)

def update_graph(option_slctd):

    dfteamMap = data.dfteamMap[data.dfteamMap['Team'] == option_slctd]

    figPie = px.pie(
    data_frame=dfteamMap,
    values='count',
    names='Nationality',

    )
    figPie.update_traces(textposition='inside', textinfo='percent+label')
    return figPie
```

Kuva 19. Ympyräkaavion vastakutsu.



Kuva 20. Ympyräkaavio.

5 YHTEENVETO

Opinnäytetyön aihe syntyi omasta kiinnostuksesta visualisointeja kohtaan. Olen aina pitänyt erityisesti urheilutilastojen visualisointien tutkimisesta. Lisäksi halusin oppia käyttämään Pythonia.

Sovelluksen kehitys onnistui hyvin, vaikka vaikeuksiakin oli välillä. Python ja Dash olivat minulle uusia asioita, joten niiden opetteluun kului aikaa. Erityisiä haasteita alussa aiheutti vastakutsujen käyttö. Datan käsittelystä ja visualisoinnista opin myös uusia asioita.

Python sopii erinomaisesti datan visualisointisovellusten luomiseen. Plotlyn lisäksi Pythonille on saatavilla myös muita visualisointikirjastoja kuten Matplotlib, Pandas Visualization, Seaborn ja Ggplot. Plotly on yksi suosituimmista kirjastoista, joten siitä löytyy paljon dokumentaatiota verkosta. Näistä on suuri hyöty Plotlyn käyttöä opetellessa. Python soveltuu myös aloittelijoille, sillä se on helppo oppia ja hyvin monipuolinen kieli. Plotly Express-rajapinta helpottaa huomattavasti kaavioiden luomista. Suuri osa parametreistä on jo valmiiksi asetettu eikä niitä tarvitse itse lisätä, kuten esim. taustaväri ja otsikot. Kaavion tyyppiä on myös helppo vaihtaa toiseen. Pylväskaaviosta pistekaavioon muuttaminen onnistuu vaihtamalla px.bar-funktion px.scatteriksi. Muihin parametreihin ei välttämättä tarvitse koskea.

Jatkokehitysmahdollisuuksia sovellukselle on paljon. Sovellukseen voisi kehittää historiallista dataa esittävän kaavion. Liukusäätimellä voisi valita ajanjakson, jolta haluaisi verrata pelaajia tai joukkueita. Ongelmana tulisi vastaan datan vaikea saatavuus. Tulisi olla suuri tietokanta tai tiedosto, joka sisältäisi pelaajien tai joukkueiden tilastoja monien vuosien ajalta.

Sovellukseen voisi myös lisätä sivun, johon käyttäjä voi itse lisätä haluamiansa kaavioita. Käyttäjän tulisi vain valita data ja haluamansa kaaviotyyppi ja painaa nappia niin kaavio ilmestyy sivulle. Käyttäjä voisi näin luoda eräänlaisen työpöydän itselleen, jossa voisi tutkia tilastoja.

Tämän tyylisestä sovelluksesta olisi hyötyä monella eri alalla. Urheilussa tätä voisi hyödyntää joukkueen kehittämiseen. Visualisoinneista voitaisiin päätellä joukkueen heikkoudet ja sen jälkeen ryhtyä kehittämään niitä.

LÄHTEET

Gioglio, J. & Walter, E. 2014. The Power of Visual Storytelling: How to Use Visuals, Videos, and Social Media to Market Your Brand. New York. McGraw-Hill.

Hoffman, J. 2021. Pros and Cons of Data Visualization. Viitattu 21.9.2021 https://wisdomplexus.com/blogs/pros-cons-data-visualization/

Kilcommins, S. 2021. Plotly Dash — Everything You Need To Know. Viitattu 16.8.2021. https://medium.datadriveninvestor.com/plotly-dash-everything-you-need-to-know-bc09a5e45395#8bdc

Kosara, R. 2007. Visualization Criticism – The Missing Link Between Information Visualization and Art. Viitattu 16.8.2021. https://kosara.net/papers/2007/Kosara-IV-2007.pdf

Koponen, J., Hilden, J. & Vapaasalo, T. 2016. Tieto näkyväksi. Helsinki. Aalto-yliopisto.

Kuenn, A. 2013. What Makes A Great Infographic? 8 Experts Weigh In. Viitattu 17.10.2021. https://martech.org/8-experts-talk-about-making-great-infographics/

Pandas. 2021. pandas.Dataframe. Viitattu 21.9.2021. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

Plotly. 2017. Introducing Dash. Viitattu 16.8.2021. https://medium.com/plotly/introducing-dash-5ecf7191b503

Plotly. 2021a. Introduction to Dash. Viitattu 9.7.2021. https://dash.plotly.com/introduction

Plotly. 2021b. About us. Viitattu 23.8.2021. https://plotly.com/about-us/

Plotly. 2021c. Dash. Viitattu 29.9.2021. https://plotly.com/dash/

Plotly. 2021d. Basic Dash Callbacks. Viitattu 23.9.2021. https://dash.plotly.com/basic-callbacks

Plotly. 2021e. Bar Charts in Python. Viitattu 16.8.2021 https://plotly.com/python/bar-charts/

Plotly. 2021f. Scatter Plots in Python. Viitattu 23.8.2021 https://plotly.com/python/line-and-scatter/

Plotly. 2021g. 3D Scatter Plots in Python. Viitattu 23.9.2021 https://plotly.com/python/3d-scatter-plots/

Plotly. 2021h. Choropleth Maps in Python. Viitattu 17.9.2021 https://plotly.com/python/3d-scatter-plots/

Plotly. 2021i. Pie Charts in Python. Viitattu 17.9.2021 https://plotly.com/python/pie-charts/

Rahman, K. 2021. Python Data Visualization Essentials Guide. New Delhi. BPB Publications.

Ware, C. 2013. Information Visualization. Waltham. Elsevier.