**Pertemuan 3 – Visualisasi data menggunakan Streamlit**

**Tujuan pembelajaran**

* Mahasiswa mampu memahami konsep visualisasi data seperti *histogram*, *lineplot*, *pieplot*, *barplot*, *scatterplot*, *boxplot*, *heatmap*, dan lain-lain.
* Mahasiswa dapat menggunakan beberapa library visualisasi data seperti *matplotlib*, *seaborn*, dan *plotly*.
* Mahasiswa dapat menggunakan *framework* visualisasi data seperti *streamlit*.

**Studi kasus: Analisis data penjualan video games (main\_vgsales.py)**

|  |
| --- |
| # import library streamlit  import streamlit as st;    # library manipulation dataset  import pandas as pd    # library manipulation array  import numpy as np    # library data visualization  import plotly.express as px |

|  |
| --- |
| # config web streamlit  st.set\_page\_config(  page\_title="My Dasboard",  page\_icon="",  layout="wide",  initial\_sidebar\_state="auto",  menu\_items={  "Get Help": "https://www.github.com/kusin",  "Report a bug": "https://www.github.com/kusin",  "About": "### Copyright 2022 all rights reserved by Aryajaya Alamsyah"  }  ) |

|  |
| --- |
| # load dataset  dataset = pd.read\_csv("dataset/vgsales.csv")    # container-header  with st.container():    st.markdown("## Data Science - Exploratory Data Analysis of Video Games Sales")    st.markdown("- Created By. Aryajaya Alamsyah, Okt 2023 (link download on  https://www.kaggle.com/datasets/gregorut/videogamesales)")    # container-dataset  with st.container():    st.dataframe(dataset, use\_container\_width=True) |

|  |
| --- |
|  |
| Gambar x. Output program |

|  |
| --- |
| # container-sales video games  with st.container():    # load history sales video games on global region    st.error("Sum of video games sales from all regions")    df = dataset.groupby(by=['Year'])['Global\_Sales'].sum().reset\_index()    df = df.sort\_values(by="Year")      # visualization lineplot    fig = px.line(df, x="Year", y="Global\_Sales")    fig.update\_traces(      line\_color="#67001f",      line\_width=2.5,    )    fig.update\_layout(      title = "History video games sales on global region",      xaxis\_title = "Year Sales",      yaxis\_title = "Sum of Sales Video",    )    st.plotly\_chart(fig,use\_container\_width=True)      # Sum video games sales on all region    df = dataset[  ['NA\_Sales', 'EU\_Sales', 'JP\_Sales', 'Other\_Sales']  ].aggregate("sum").sort\_values(ascending=False).reset\_index()    df.columns = ["Region", "Sales"]      # visualization of sum video games sales    col1, col2 = st.columns([1,1], gap="large")    with col1:      # show pieplot      fig = px.pie(df, values='Sales', names='Region',  hole=0.25, color\_discrete\_sequence=px.colors.sequential.RdBu)      fig.update\_layout(        title = "Sales Video Games by Region",        legend=dict(orientation='h', yanchor='top', y=0.0, xanchor='center', x=0.5),        showlegend=True,      )      st.plotly\_chart(fig,use\_container\_width=True)    with col2:      # show barplot      fig = px.bar(        df, x="Region", y="Sales", text\_auto='.3s'      )      fig.update\_traces(        marker\_color = px.colors.diverging.RdYlBu      )      fig.update\_layout(        title = "Sales Video Games by Region",        xaxis\_title = "Region Sales",        yaxis\_title = "Sum of Sales Video",      )      st.plotly\_chart(fig, use\_container\_width=True) |

|  |
| --- |
|  |
|  |
| Gambar x. Output program |

|  |
| --- |
| # container-best game, platform, genre, publisher  with st.container():    # labels    st.error("Best of Game, Platform, Genre, Publisher on All Region")      # func groupBarplot    def barplot(column, title, xlabel, ylabel):        # grouping on pandas      sales\_by\_column = dataset.groupby(column)[  ['NA\_Sales', 'EU\_Sales', 'JP\_Sales', 'Other\_Sales', 'Global\_Sales']  ].aggregate("sum").sort\_values(by=['Global\_Sales'],  ascending=[False]).head(5).reset\_index()        # show barplot      fig = px.bar(sales\_by\_column, barmode='group',  x=column, y=['NA\_Sales', 'EU\_Sales', 'JP\_Sales', 'Other\_Sales'])        colors = {        'NA\_Sales': 'rgb(165,0,38)',        'EU\_Sales': 'rgb(215,48,39)',        'JP\_Sales': 'rgb(244,109,67)',        'Other\_Sales': 'rgb(253,174,97)'      }      for i, col in enumerate(['NA\_Sales', 'EU\_Sales', 'JP\_Sales', 'Other\_Sales']):        fig.update\_traces(marker\_color=colors[col],  selector=dict(name=col),  name=f'{col.split("\_")[0]} Sales')        fig.update\_layout(        title=title,        xaxis\_title=xlabel,        yaxis\_title=ylabel,        xaxis=dict(tickangle=0),        yaxis=dict(tickangle=0),        legend=dict(orientation='h', yanchor='top', y=1.05, xanchor='center', x=0.5)      )      return fig      # visualization of best game, platform, genre, publisher    col1, col2 = st.columns(2, gap="large")    with col1:      col1.plotly\_chart(        barplot(          'Name',          'Top 5 of game on all regions',          'Game name',          'Sum of Video Games'        ), use\_container\_width=True      )    with col2:      col2.plotly\_chart(        barplot(          'Platform',          'Top 5 of platform on all regions',          'Platform name',          'Sum of Video Games'        ), use\_container\_width=True      )      # visualization of best game, platform, genre, publisher    col1, col2 = st.columns(2, gap="large")    with col1:      col1.plotly\_chart(        barplot(          'Genre',          'Top 5 of genre on all regions',          'Genre name',          'Sum of Video Games'        ), use\_container\_width=True      )    with col2:      col2.plotly\_chart(        barplot(          'Publisher',          'Top 5 of publisher on all regions',          'Publisher name',          'Sum of Video Games'        ), use\_container\_width=True      ) |

|  |
| --- |
|  |
|  |
| Gambar x. Output program |

**Studi kasus: Visualisasi data covid-19 di Indonesia (main\_covid19.py)**

|  |
| --- |
| # import library streamlit  import streamlit as st;    # library manipulation dataset  import pandas as pd    # library manipulation array  import numpy as np    # library data visualization  import plotly.express as px  import plotly.graph\_objects as go |

|  |
| --- |
| # config web streamlit  st.set\_page\_config(  page\_title="My Dasboard – Covid 19",  page\_icon="",  layout="wide",  initial\_sidebar\_state="auto",  menu\_items={  "Get Help": "https://www.github.com/kusin",  "Report a bug": "https://www.github.com/kusin",  "About": "### Copyright 2022 all rights reserved by Aryajaya Alamsyah"  }  ) |

|  |
| --- |
| # func load dataset  @st.cache\_data  def load\_xlsx(file\_name,sheet\_name):      # load dataset covid-19    df = pd.read\_excel("../dataset/"+file\_name, sheet\_name=sheet\_name)      # convert obj or str to datetime    df["date"] = pd.to\_datetime(df["date"], format="%Y-%m-%d")      # setting date    df["date"] = df['date'].dt.date      # setting columns    df = df[[          "date", "cumulative\_positive", "cumulative\_recovery", "cumulative\_dead",          "daily\_positive", "daily\_recovery", "daily\_dead"        ]]      # sorting dataset by date desc    df = df.sort\_values(by="date", ascending=False)      # return values    return df |

|  |
| --- |
| # container-header-fuild  with st.container():    st.markdown("## Data Visualization of Covid-19 in Indonesia")    # container-dataset  dataset = load\_xlsx("dataset\_covid.xlsx", "data-covid-indonesia")  with st.container():    # split two columns    col1, col2 = st.columns([0.65,0.35], gap="small")    # show dataset    with col1:      st.dataframe(data=dataset, use\_container\_width=True, hide\_index=True)    # summary statistics    with col2:      # calculate recovery and date      recovery = (        dataset["daily\_recovery"].sum() / dataset["daily\_positive"].sum()      )\*100      dead = (        dataset["daily\_dead"].sum() / dataset["daily\_positive"].sum()      )\*100      # ----------------------------------------------------------------------------      # split two columns      sub\_col1, sub\_col2 = st.columns(2)      sub\_col1.metric(        label="Percentage recovery",        value="{:.2f}".format(recovery)+"%",        delta="0,35%",      )      sub\_col2.metric(        label="Percentage dead",        value="{:.2f}".format(dead)+"%",        delta="0,00%",      )      # ----------------------------------------------------------------------------      # split three columns      sub\_col1, sub\_col2, sub\_col3 = st.columns(3)      sub\_col1.metric(        label="Cumulative Positive",        value="{:,}".format(dataset["daily\_positive"].sum()),        delta="3,373 People"      )      sub\_col2.metric(        label="Cumulative Recovery",        value="{:,}".format(dataset["daily\_recovery"].sum()),        delta="3.919 People"      )      sub\_col3.metric(        label="Cumulative Dead",        value="{:,}".format(dataset["daily\_dead"].sum()),        delta="106 People"      )      # ----------------------------------------------------------------------------      # split three columns      sub\_col1, sub\_col2, sub\_col3 = st.columns(3)      sub\_col1.metric(        label="Daily Positive",        value="{:,}".format(dataset["daily\_positive"].iloc[0]),        delta="-732 People"      )      sub\_col2.metric(        label="Daily Recovery",        value="{:,}".format(dataset["daily\_recovery"].iloc[0]),        delta="187 People"      )      sub\_col3.metric(        label="Daily Dead",        value="{:,}".format(dataset["daily\_dead"].iloc[0]),        delta="26 People"      )      # ---------------------------------------------------------------------------- |

|  |
| --- |
|  |
| Gambar x. Output program |

|  |
| --- |
| # container-lineplot  with st.container():      # split two columns    col1, col2 = st.columns([0.5,0.5], gap="medium")      # col1 - cumulative data of positive, recovery, and dead    with col1:      fig = go.Figure()      fig.add\_trace(        go.Scatter(          x=dataset["date"],          y=dataset["cumulative\_positive"],          name="Cumulative Positive",          line=dict(color="blue",width=2),        )      )      fig.add\_trace(        go.Scatter(          x=dataset["date"],          y=dataset["cumulative\_recovery"],          name="Cumulative Recovery",          line=dict(color="green",width=2),        )      )      fig.add\_trace(        go.Scatter(          x=dataset["date"],          y=dataset["cumulative\_dead"],          name="Cumulative Dead",          line=dict(color="red",width=2),        )      )      fig.update\_layout(        title="TimeSeries of cumulative positive, recovery, and dead",        xaxis\_title="",        yaxis\_title="Number of cases",        xaxis=dict(tickangle=0),        yaxis=dict(tickangle=0),        legend=dict(          title='', orientation='h',          yanchor='top', y=1.05,          xanchor='center', x=0.5        )      )      st.plotly\_chart(fig, use\_container\_width=True)      # -----------------------------------------------------------------------------      # col2 - daily data of positive, recovery, and dead    with col2:      fig = go.Figure()      fig.add\_trace(        go.Scatter(          x=dataset["date"],          y=dataset["daily\_positive"],          name="Daily Positive",          line=dict(color="blue",width=2),        )      )      fig.add\_trace(        go.Scatter(          x=dataset["date"],          y=dataset["daily\_recovery"],          name="Daily Recovery",          line=dict(color="green",width=2),        )      )      fig.add\_trace(        go.Scatter(          x=dataset["date"],          y=dataset["daily\_dead"],          name="Daily Dead",          line=dict(color="red",width=2),        )      )      fig.update\_layout(        title="TimeSeries of daily positive, recovery, and dead",        xaxis\_title="",        yaxis\_title="Number of cases",        xaxis=dict(tickangle=0),        yaxis=dict(tickangle=0),        legend=dict(          title='', orientation='h',          yanchor='top', y=1.05,          xanchor='center', x=0.5        )      )      st.plotly\_chart(fig, use\_container\_width=True)      # ----------------------------------------------------------------------------- |

|  |
| --- |
|  |
| Gambar x. Output program |

|  |
| --- |
| # container-scatterplot  with st.container():      # split two columns    col1, col2 = st.columns([0.5,0.5], gap="medium")      # col1 - daily data of positive vs recovery    with col1:      fig = go.Figure()      fig.add\_trace(        go.Scatter(          x=dataset["daily\_positive"],          y=dataset["daily\_recovery"],          text=dataset["date"],          mode="markers",        )      )      fig.update\_traces(        marker={          "size" : 12,          "color": "green",          "opacity": 0.75,          "line": {"width": 0.5, "color": "black"},          "symbol": "circle"        }      )      fig.update\_layout(        title="Scatter plot of positive and recovery",        xaxis\_title="Daily positive",        yaxis\_title="Daily recovery",        xaxis=dict(tickangle=0),        yaxis=dict(tickangle=0),      )      st.plotly\_chart(fig, use\_container\_width=True)      # ----------------------------------------------------------------------------      # col1 - daily data of positive vs dead    with col2:      fig = go.Figure()      fig.add\_trace(        go.Scatter(          x=dataset["daily\_positive"],          y=dataset["daily\_dead"],          text=dataset["date"],          mode="markers",        )      )      fig.update\_traces(        marker={          "size" : 12,          "color": "red",          "opacity": 0.75,          "line": {"width": 0.5, "color": "black"},          "symbol": "circle"        }      )      fig.update\_layout(        title="Scatter plot of positive and dead",        xaxis\_title="Daily positive",        yaxis\_title="Daily dead",        xaxis=dict(tickangle=0),        yaxis=dict(tickangle=0),      )      st.plotly\_chart(fig, use\_container\_width=True)      # ---------------------------------------------------------------------------- |

|  |
| --- |
|  |
| Gambar x. Output program |

|  |
| --- |
| # -------------------------------------------------------------------------------#  # data acquisition province ---------------------------------------------------- #  # -------------------------------------------------------------------------------#  dataset = pd.read\_excel(  "../dataset/dataset\_covid.xlsx", sheet\_name="data-covid-provinsi",  engine="openpyxl"  )  st.info("Summary statistic on each province") |

|  |
| --- |
| # container-positive  with st.container():      # sorting sum-positive with most cases    df\_positive = dataset.sort\_values("sum\_positive", ascending=False)    df\_positive = df\_positive[["province", "sum\_positive"]].head(5)      # split two columns    col1, col2 = st.columns([0.5,0.5], gap="medium")      # col1 - Pieplot (Top 5 province of sum positive)    with col1:      fig = px.pie(        df\_positive, names="province", values="sum\_positive", hole=0.5,      )      fig.update\_traces(        textinfo="percent+value",        textfont\_size=14,        marker\_colors = px.colors.qualitative.Pastel,        marker\_line\_color="#FFFFFF",        marker\_line\_width=1.25,      )      fig.update\_layout(        title = "Top 5 province of sum positive",        legend=dict(orientation='h', x=0.05, y=0.0),      )      st.plotly\_chart(fig, use\_container\_width=True)  # ----------------------------------------------------------------------------      # col2 - Barplot (Top 5 province of sum positive)    with col2:      fig = px.bar(        df\_positive, x="province", y="sum\_positive", text\_auto=True,      )      fig.update\_traces(        textfont\_size=14,        textangle=0,        textposition="inside",        marker\_color="#0071c5",        marker\_line\_color="#292929",        marker\_line\_width=1,      )      fig.update\_layout(        title = "Top 5 province of sum positive",        xaxis\_title = "province",        yaxis\_title = "sum positive",      )      st.plotly\_chart(fig, use\_container\_width=True)  # ---------------------------------------------------------------------------- |

|  |
| --- |
|  |
| Gambar x. Output program |

|  |
| --- |
| # container-recovery  with st.container():      # sorting sum-recovery with most cases    df\_recovery = dataset.sort\_values("sum\_recovery", ascending=False)    df\_recovery = df\_recovery[["province", "sum\_recovery"]].head(5)      # split two columns    col1, col2 = st.columns([0.5,0.5], gap="medium")      # col1 - Pieplot (Top 5 province of sum recovery)    with col1:      fig = px.pie(        df\_recovery, names="province", values="sum\_recovery", hole=0.5,      )      fig.update\_traces(        textinfo="percent+value",        textfont\_size=14,        marker\_colors = px.colors.qualitative.Pastel,        marker\_line\_color="#FFFFFF",        marker\_line\_width=1.25,      )      fig.update\_layout(        title = "Top 5 province of sum recovery",        legend=dict(orientation='h', x=0.05, y=0.0),      )      st.plotly\_chart(fig, use\_container\_width=True)  # ----------------------------------------------------------------------------      # col2 - Barplot (Top 5 province of sum recovery)    with col2:      fig = px.bar(        df\_recovery, x="province", y="sum\_recovery", text\_auto=True,      )      fig.update\_traces(        textfont\_size=14,        textfont\_color="black",        textangle=0,        textposition="inside",        marker\_color="#63c689",        #marker\_color="green",        marker\_line\_color="#292929",        marker\_line\_width=0.5,      )      fig.update\_layout(        title = "Top 5 province of sum recovery",        xaxis\_title = "province",        yaxis\_title = "sum recovery",      )      st.plotly\_chart(fig, use\_container\_width=True)  # ---------------------------------------------------------------------------- |

|  |
| --- |
|  |
| Gambar x. Output program |

|  |
| --- |
| # container-dead  with st.container():      # sorting sum-dead with most cases    df\_dead = dataset.sort\_values("sum\_dead", ascending=False)    df\_dead = df\_dead[["province", "sum\_dead"]].head(5)      # split two columns    col1, col2 = st.columns([0.5,0.5], gap="medium")      # col1 - Pieplot (Top 5 province of sum dead)    with col1:      fig = px.pie(        df\_dead, names="province", values="sum\_dead", hole=0.5,      )      fig.update\_traces(        textinfo="percent+value",        textfont\_size=14,        marker\_colors = px.colors.qualitative.Pastel,        marker\_line\_color="#FFFFFF",        marker\_line\_width=1.25,      )      fig.update\_layout(        title = "Top 5 province of sum dead",        legend=dict(orientation='h', x=0.05, y=0.0),      )      st.plotly\_chart(fig, use\_container\_width=True)     # -----------------------------------------------------------------------------    # col2 - Barplot (Top 5 province of sum dead)    with col2:      fig = px.bar(        df\_dead, x="province", y="sum\_dead", text\_auto=True,      )      fig.update\_traces(        textfont\_size=14,        textfont\_color="black",        textangle=0,        textposition="inside",        marker\_color="#dc143c",        marker\_line\_color="#292929",        marker\_line\_width=0.5,      )      fig.update\_layout(        title = "Top 5 province of sum dead",        xaxis\_title = "province",        yaxis\_title = "sum dead",      )      st.plotly\_chart(fig, use\_container\_width=True)      # -----------------------------------------------------------------------------    # container footer  with st.container():    st.info(  "Copyright all rights reserved 2024 by Aryajaya Alamsyah, S.Kom., M.Kom., MTA."  ) |

|  |
| --- |
|  |
| Gambar x. Output program |

**Selesai, Selamat Mencoba :3**