COMP1005 Week 9 Cheat Sheet

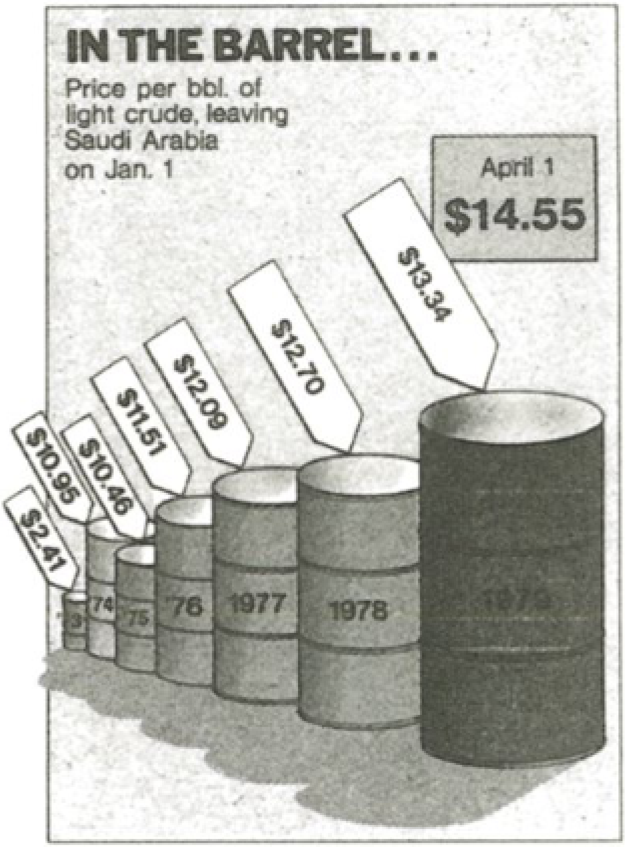
Lisa Luff

10/14/2020

Contents

Visualisation

- Great resource is “The Visual Display of Quantitative Information” by Tufte
- Important to present data in a clear, unbiased way to avoid being misleading
- Graphical Integrity:
  - How accurately the visual elements represent the data
- Tuftes Principles of Graphical Integrity:
  1. The representation in numbers, as physically measured on the surface of the graphic itself, should be directly proportional to the numerical quantities measured
  - “The lie factor”:
    * lie factor = $\frac{\text{size of effect shown in graphic}}{\text{size of effect in data}}$
    * This should be 1, but can often be between 2 and 5



Lie Factor 9.5... by volume 59.4

Figure 1: “Tufte 1”

2. Clear, detailed, and thorough labeling should be used to defeat graphical distortion and ambiguity

- Explain the data on the graphic itself
- Label important events in the data

3. Show data variation, not design variation

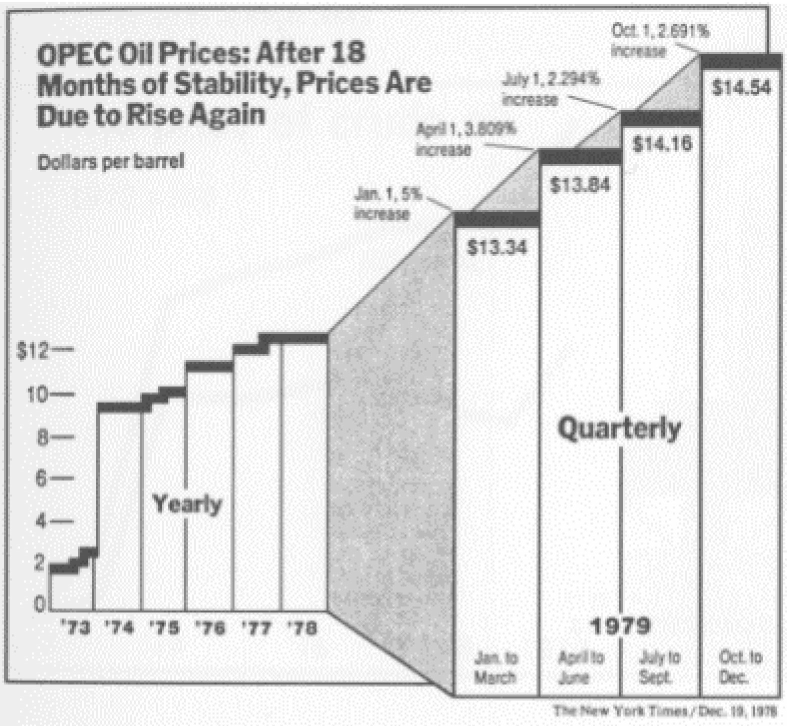


Figure 2: "Tufte 3"

4. In time-series displays of money, deflated and standarised units of monetary measurement are nearly always better than nominal units

- Well known units are best
- Seasonally adjusted might be better
- Might need to normalise data if over a long time span

5. The number of information-carrying (variable) dimensions depicted should not exceed the number of dimensions in the data

- Graphics must not quote data out of context

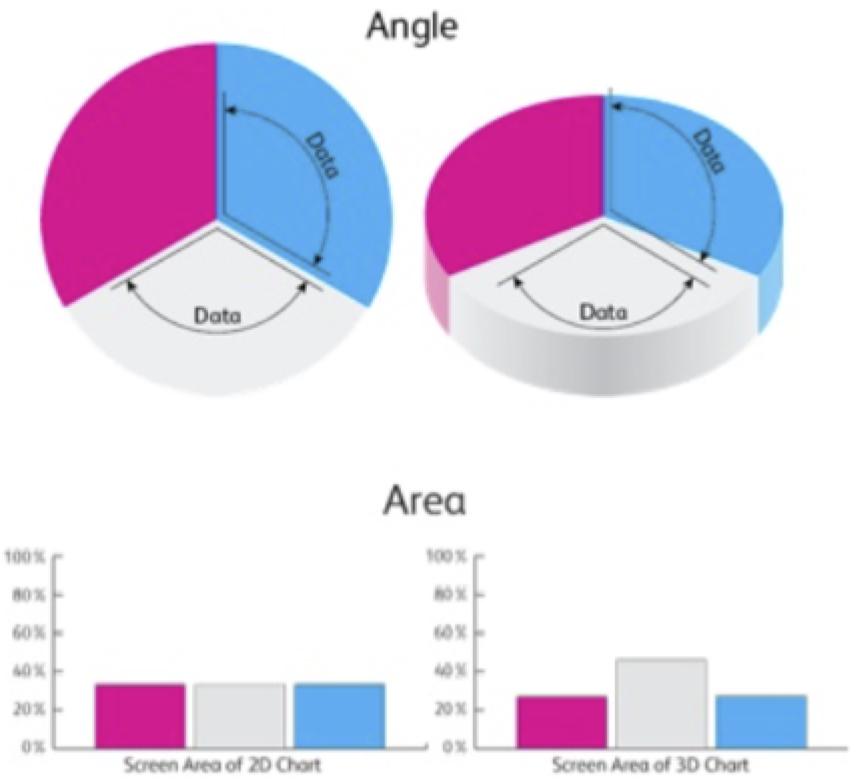


Figure 3: "Tufte 5"

- Achieving visual accuracy:

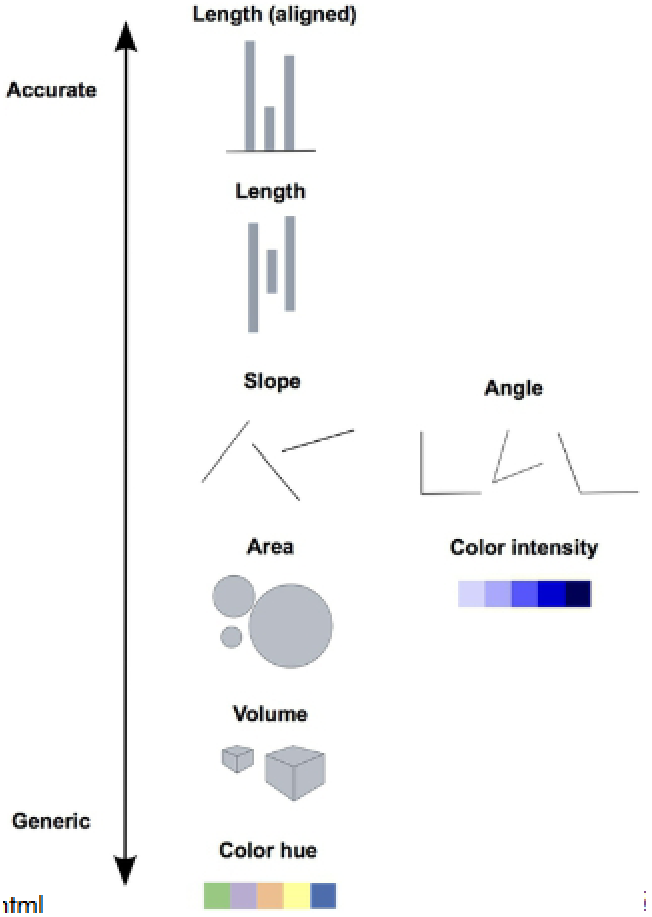


Figure 4: “Visual Accuracy”

- Choosing representation

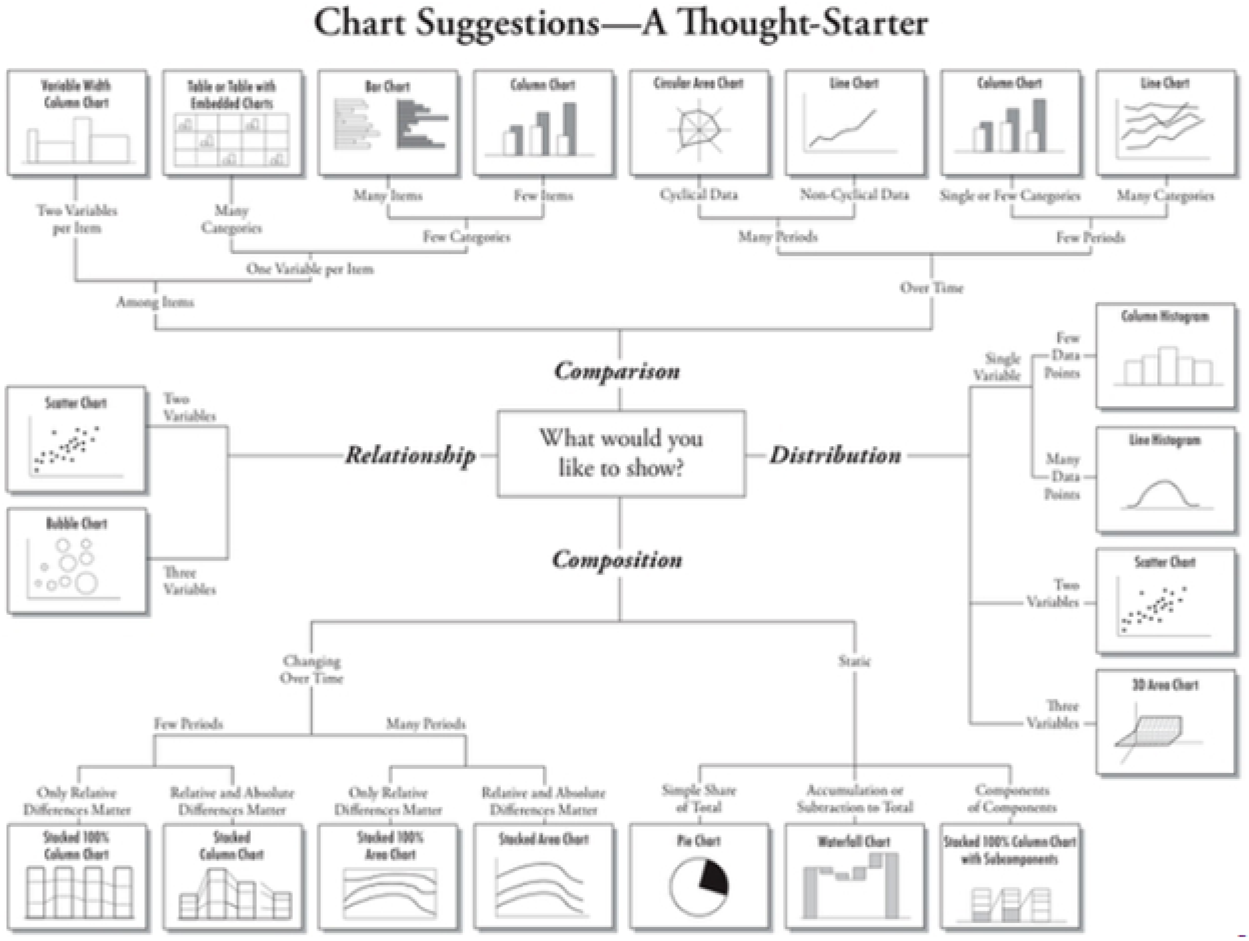


Figure 5: “Representation Flow Chart”

Python Plotting Packages

- Matplotlib
  - Has styles, which looks nice. Good enough for publication-quality plots
- Pandas
  - Good for simple plots, but needs matplotlib knowledge to customise
- Seaborn
  - Supports more complex visualisation, but requires matplotlib knowledge to customise. Colour schemes are nice
- ggplot
  - Lots of promise, but has growing pains
- bokeh
  - Robust tool to do more visualisation using a visualisation server
- pygal
  - Only one able to generate interactive svg graphs and png files. Not as flexible as matplotlib based options
- Plotly
  - Most interactive graphs. Can save them offline, and create very rich web-based visualistions

Simple Graphing

- Matplotlib -
  - import matplotlib.pyplot as plt
    - *plotv* = *datav*.plot(kind = ‘bar’)
  - How to improve on this -
    - *plotv* = *datav*.sort(columns = ‘*colname*’, ascending = False).plot(kind = ‘bar’, legend = None, title = “*title*”)
    - → *plotv*.set_xlabel(“*labelx*”)
    - → *plotv*.set_ylabel(“*ylabel*”)
  - print(plt.style.available) will show you all the style options
  - The styles will apply to all future plots if you don’t use it within a with statement:
    - with plt.style.context((‘*style*’))
    - → plt.plot(*datav*)

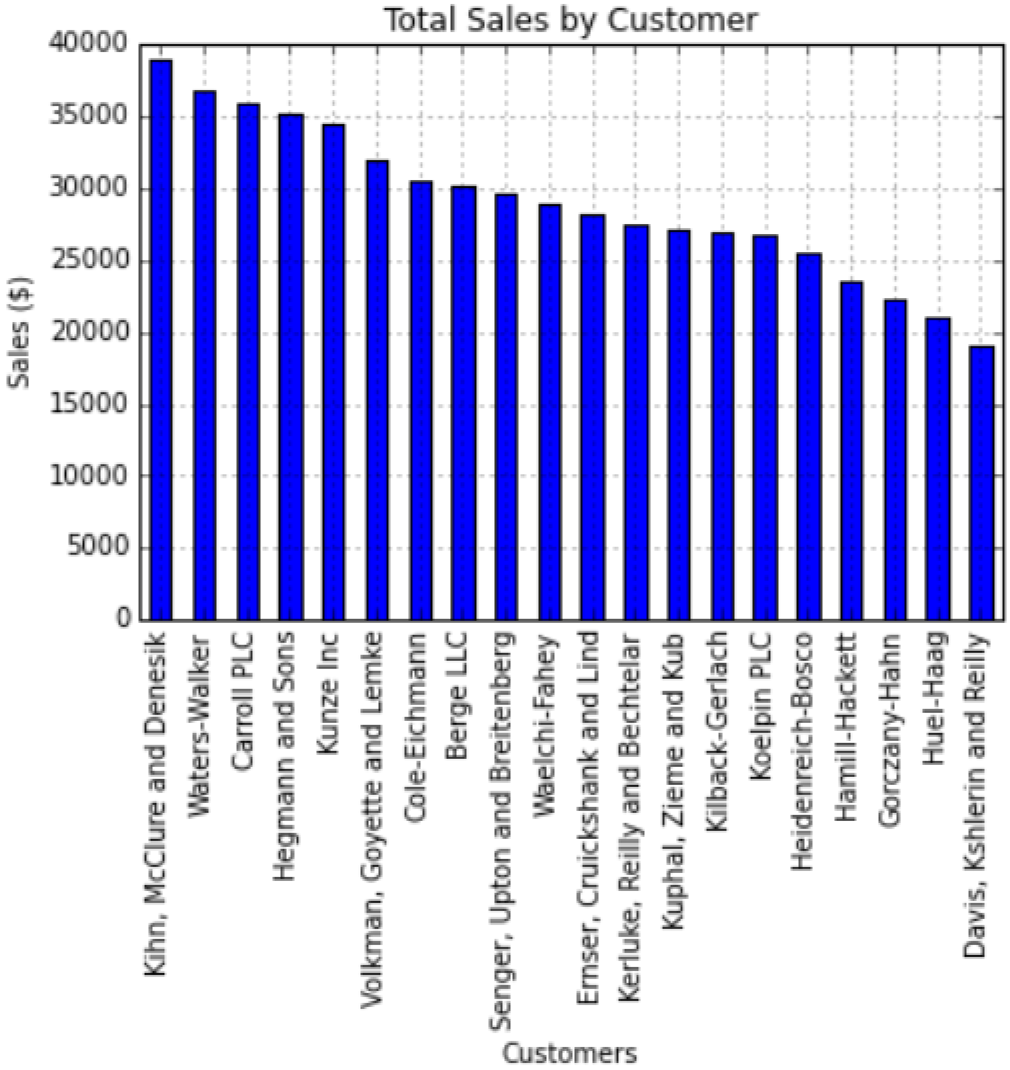


Figure 6: “Matplotlib.pyplot graph”

- Pandas
  - import matplotlib.pyplot as plt
  - import pandas as pd
  - Builds on matplotlib
    - plt.style.use = ‘default’ (also in matplotlib)
    - → *plotv* = *datav*.plot(kind = “bar”, x = *datav*[“*colname*”]), title = “*title*”, legend = False)
    - → fig = *plotv*.get_figure()
    - → fig.savefig(“*figname.filetype*”)

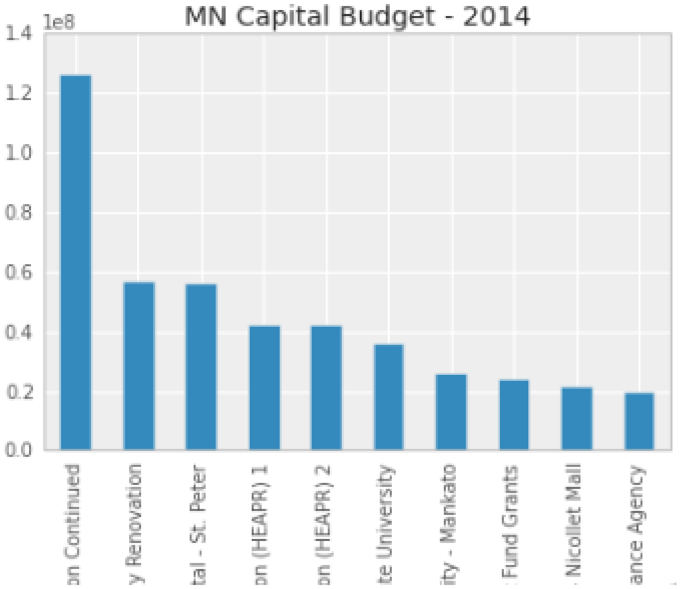


Figure 7: “Pandas graph”

- Seaborn
  - import matplotlib.pyplot as plt
  - import pandas as pd
  - import seaborn as sns
  - Can use plotting routines and styles to present dataframes
  - Based on matplotlib
    - *datav* = pd.read_csv(“*file.csv*”)
    - → *datav* = *datav*.sort_values(by = ‘*colname*’, ascending = False)[:10]
    - → sns.set_style(“*style*”)
    - → *plotv* = sns.barplot(x = *datav*[“*colname*”], y = *datav*[“*colname*”], palette = “*palette*”, order = *datav*[“*colname*”].tolist())
    - → plt.xsticks(rotation = 90) (makes some lines behind)
    - → plt.show()

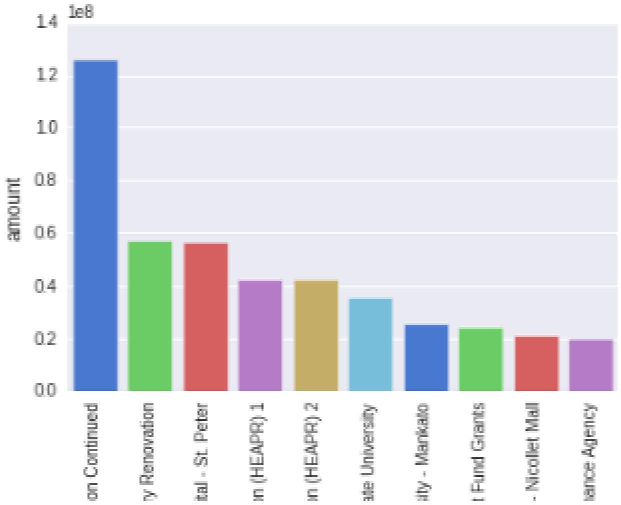


Figure 8: “Seaborn graph”

- More seaborn
  - sns.set(style = “*style*”)
  - → f, axes = plt.subplots(3, 3, figsize = (9, 9), sharex = True, shary = True)
  - → for ax, s in zip(axes.flat, np.linspace(0, 3, 10)): (Rotates the start point around a cubehelix hue circle)
  - →→ cmap = sns.cubehelix_palette(start = s, light = 1, as_cmap = True) (creates the cubehelix colourmap)
  - →→ x, y = *data*
  - →→ sns.kdeplot(x, y, cmap = cmap, shade = True, cut = 5, ax = ax)
  - →→ ax.set(xlim = (-3, 3), ylim = (-3, 3))
  - →→ f.tight_layout()

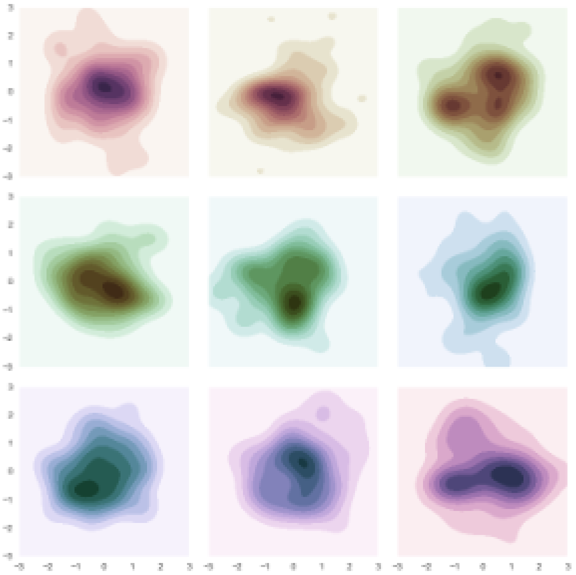


Figure 9: “Fancy Seaborn graph”

- Bokeh
- **bokeh.chart no longer exists, see Python cheat sheet for how to use bokeh.plotting**
  - import pandas
  - from bokeh.charts import Bar
  - Not based on matplotlib
  - Focused on web-visualisations
  - Images inline or in separate webpage
    - *colname* = *datav*[*colname*].values.tolist()
    - → *col2name* = list(*datav*[“*col2name*”].astype(float).values)
    - → *plotv* = Bar(*col2name*, *colname*, filename = “*filename.html*”) (might not work?)
    - → *plotv*.title(“*title*”)
    - → *plotv*.xlabel(“*labelx*”).ylabel(“*labely*”)
    - → *plov*.show()

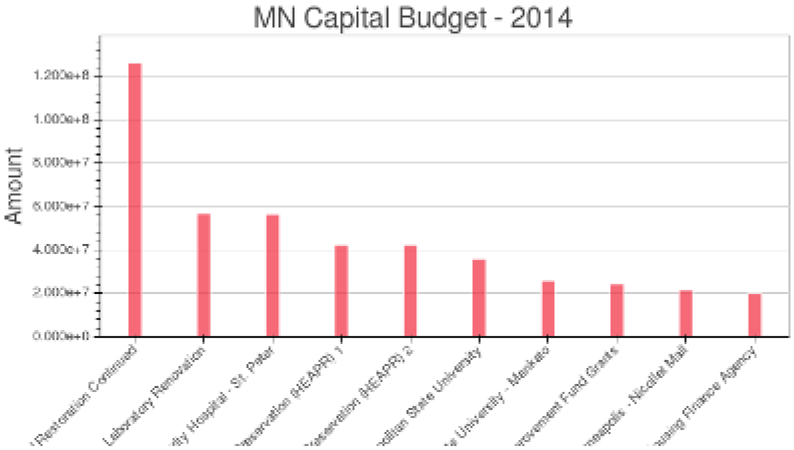


Figure 10: “Bokeh graph”

- More Bokeh
- import pandas as pd
- from bokeh.charts import output_file, Chord
- from bokeh.io import show
- from bokeh.sampledata.les_mis import data
  - nodes = *datav*[‘*datanodesv*’]
  - → links = *datav*[‘*datalinksv*’]
  - → nodes_df = pd.DataFrame(*datanodesv*)
  - → links_df = pd.DataFrame(*datalinksv*)
  - → source_data = links_df.merge(nodes_df, how = ‘left’, left_on = ‘source’, right_index = True)
  - → source_data = source_data.merge(nodes_df, how = ‘left’, left_on = ‘target’, right_index = True)
  - → source_data = source_data[source_data[“*colname*”] > 5]
  - → chord_from_df = Chord(source_data, source = “*colxname*”, target = “*colyname*”, value = “*colname*”)
  - → output_file(‘chord_from_df.html’, mode = “inline”)
  - → show(chord_from_df)

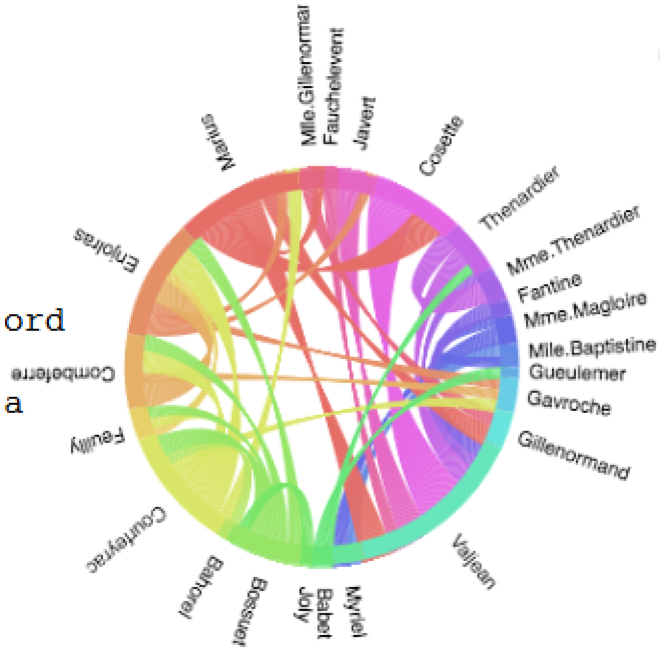


Figure 11: “More Bokeh”

- Holoviews with Bokeh
- import holoviews as hv
- import numpy as np
- hv.extension(‘bokeh’, ‘matplotlib’)
- Using Holoviews with Bokeh
  - *tablev* = hv.Table((*xdatav*, *ydatav*, *zdatav*), kdims = [‘x’, ‘y’], vdims = [‘z’])
  - → hv.BoxWhisker(*table*)

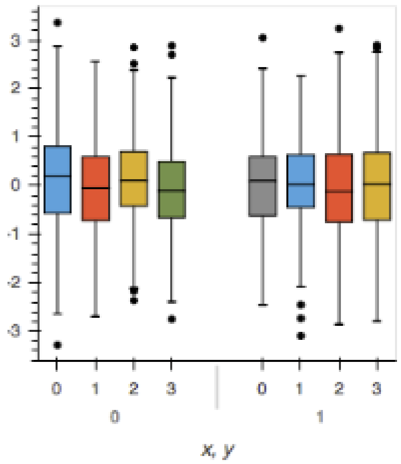


Figure 12: “Holoview with Bokeh graph”