

Summer term 2025

## Visual Data Analysis

### Assignment Sheet 7

Solution has to be submitted via eCampus by May 23, 2025, 10:00 a.m.

If you have questions concerning the exercises, please use the forum on eCampus.

- Please work on this exercise in **small groups** of 3 students. Submit each solution only once, but clearly indicate who contributed to it by forming a team in eCampus. Remember that all team members have to be able to explain all answers.
- Please submit your answers in PDF format, and your scripts as \*.py/\*.ipynb files. If you are using [Jupyter notebook](#), please also export your scripts and results as PDF.

### Exercise 1 (Graph Visualization in Python, 15 Points)

In this exercise you will learn how to visualize graphs in Python. We suggest to further use plotly and the [Networkx](#) package, which was used for our own example solution. In case you should already know and prefer an alternative Python package that allows you to perform the same tasks, you are free to use it.

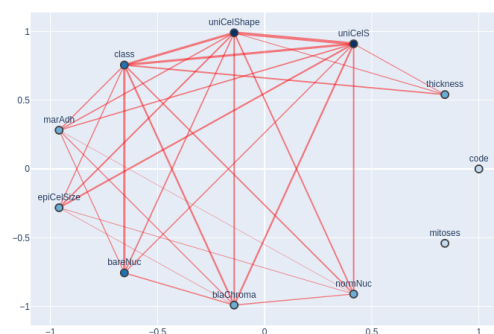


Figure 1: Variable correlation graph.

- a) Read the [Breast Cancer Dataset](#) `breast-cancer-wisconsin.xlsx` (from Sheet 3) and fill in the missing values as before. Then compute the Pearson correlation between any pair of variables, and store them in a matrix. (2P)
- b) Create a graph from the correlation matrix and visualize it with a force-directed layout. Represent each variable as a node in the graph. Insert an edge between two variables whenever the Pearson correlation between them exceeds the threshold  $\rho > 0.6$ . Tune the hyperparameters of the force-directed layout, such that the disconnected nodes do not drift too far away. (5P)

- c) Modify the visual attributes of edges to reflect the magnitude of the correlation. (2P)
- d) Produce an alternative visualization with a circular layout. Color the nodes so that there are four sets of nodes, one color for having at least one correlation more than 0.9 to other nodes, another for having at least a correlation  $0.8 < \rho_{\max} \leq 0.9$ , one for having a correlation  $0.6 < \rho_{\max} \leq 0.8$  and the last one for the remaining nodes. (3P)
- e) Answer the following questions:
  - At the selected threshold, which nodes are disconnected from the rest of the graph and what do they indicate? (1P)
  - If two nodes A and B are strongly correlated, and node C is strongly correlated with node B, can we conclude that node C will be also strongly correlated with node A? (1P)
  - Based on the visualization, which variables would you propose to predict the class? (1P)

## Exercise 2 (Large-Scale Graph Visualization, 10 Points)

In the lecture, we stated that node-and-link diagrams do not scale well to large graphs, and briefly mentioned that the Zoomable Adjacency Matrix Explorer (ZAME) has been proposed as an alternative. In this task, we invite you to learn more about this approach from the corresponding paper `elmqvist-zame-2008.pdf`.

Please answer the following questions in your own words. Remember that **we will not grant even partial credit for copy-pasted text**.

- a) What is the difference between geometric zoom and detail zoom in the system? (1P)
- b) The paper suggests eight different ways to visualize aggregated edge information. Which of them are actually used in the screenshots in Figures 1 and 9? (2P)
- c) What is the role of tile management within the ZAME system? What is an LRU cache? (2P)
- d) How does ZAME aggregate nominal attributes? Why is that problematic? (1P)
- e) Why is the Traveling Salesman Problem relevant to adjacency matrix based graph visualization? (2P)
- f) In the pseudocode listed in the paper's Figure 4, some modifications are highlighted in boldface, on lines starting with a bar. What is the purpose of these modifications? (2P)

## Exercise 3 (Interactive Visualization with Dash, 25 Points)

Interaction is a key component of many visualizations. One tool for building interactive visualizations in the form of web applications is the Dash framework. The goal of this task, and a follow-up task on the next sheet, is to learn its basics. You should find the Dash tutorial, as well as the documentation chapter on Pattern Matching Callbacks, useful for solving this exercise (<https://dash.plotly.com/>).

- a) Read the file `Data_Cortex_Nuclear.xls` (from Sheet 3), and restrict it to the classes t-CS-s and c-CS-s. Run PCA, ISOMAP and t-SNE as dimensionality reduction techniques. Within the Dash framework, create a single scatter plot that will show the output from one of the techniques, as selected by the user. Add a **Radio** component from Dash to switch between the different techniques. A **callback** function is a function which is triggered by an event such as a mouse click, double click, or selection. In Dash, you can use `@app.callback` to define your event callback, with the dropdown **value** as the input. (10P)

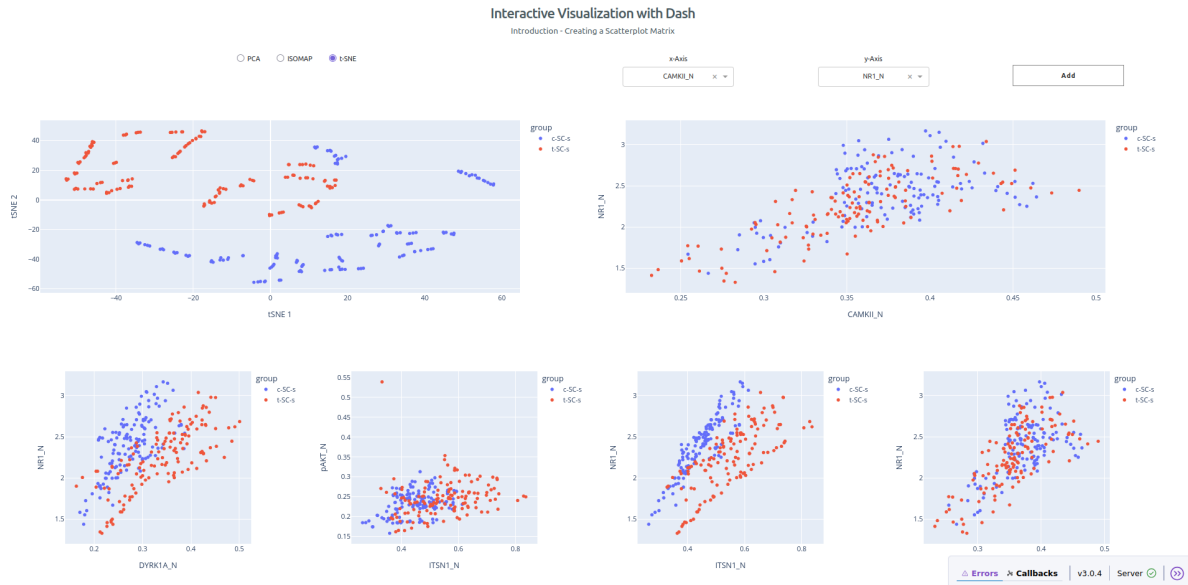


Figure 2: A screenshot of the interactive visualization you are asked to implement in this exercise.

- Add a second scatter plot with corresponding **dropdown** menus that should give you the opportunity to map any individual feature (i.e., protein) to any of the axes. (5P)
- Allow the user to show up to four scatterplots by implementing an “Add” button. Clicking on it should replicate the scatterplot that is on the right side of your previously implemented app, keeping the current user-defined axis settings. The new plots should appear below the old ones, as in Figure 2. To maintain a clean interface, adding a fifth plot should remove the oldest plot. (10P)

*Hints:*

- You can customize your app’s appearance using CSS and the `dash_bootstrap_components` module. If you use `plotly.express`, try:

```
external_stylesheets = [
    'https://codepen.io/chridyp/pen/bWLwgP.css',
    dbc.themes.BOOTSTRAP,
    'https://use.fontawesome.com/releases/v5.12.1/css/all.css'
]
```

- If you haven’t looked at States yet - now is the time to do so. See the Basic Callbacks Tutorial.
- All the components in a Div container are stored in its “children” list. (This assignment usually happens automatically when you define your layout.) You can achieve a flexible layout by giving the Div container a component ID and using its “children” list as both a State and an Output for your callback.
- If the callback was triggered by the Add button, you want to append the new plot to the children list of the parent Div.

**Good Luck!**