Projects
Visual Analytics

Summer semester 2025
Project 1

This project will need to be uploaded in ILIAS before **23.05.2025** at **18:00 o'clock**.
The presentation will be done during your tutorial in the week of **27.05.2025**.

- Exercises will be done in groups of 3-4 students. **Every** group member needs to be present at the presentation.
- All students should know the reasoning behind design decisions and are able to articulate this during the presentation.
- Additional libraries/dependencies/frameworks are not permitted.
- 25 out of 40 points must be obtained over the 2 exercises in order to be allowed to take the exam.
- At least one person per group needs to upload the presentation and the code used to generate the visualizations in Ilias before 18:00 23.05.2025. Failure to upload will result in 0 points for this exercise.

Task 1: Setup

For all exercises in the Visual Analytics course, we will be using the programming language *JavaScript*, one of the main programming languages for web applications and Visual Analytics in general. By developing web-based applications we can easily create cross-platforms dashboards for visual exploration and analysis of complex data running on every modern device using the web browser.

While this course is not about web development in general, the programming language used in the assignment will be JavaScript and hence some general information is required. In Task 1 we will guide you through setting up a basic client-server application using JavaScript. For this course, we suggest using the *VSCode* (Visual Studio Code) editor for developing with JavaScript (https://code.visualstudio.com/ - *not to be confused with Visual Studio*). We recommend installing the following additional plugins that will help during the exercises of this course. Note that using the editor or these plugins is not mandatory, but using JavaScript is.

- Auto Rename Tag (formulahendry.auto-rename-tag)
- D3.js Snippets (Hridoy.d3-js-snippets)
- ESLint (dbaeumer.vscode-eslint)

Task 1.1 Template

For the two project, we have provided you with a template to get a quick start. In the following sections, we will setup and initialize this template. Each person in the group should ensure this they can run the basic template.

Task 1.1.1 NodeJS

The template is written in NodeJS, which you can download via the instructions below.

NodeJS allows us to run JavaScript outside the web browser. Similar to *Python*, with NodeJS JavaScript files can be run in a terminal (cmd.exe on Windows). This allows us to create a webserver for data pre-processing and a client-interface with the same programming language. More details can be found online, for example at https://www.w3schools.com/nodejs/.

1. Install NodeJS
   - You can use the newest version, but if something is not working as expected, try falling back to the latest lts-release **20.x** https://nodejs.org/en/
   - For installation you **can** use *NVM* instead of directly installing NodeJS
     (NVM allows easy switching between multiple NodeJS-Versions that are installed in parallel)
     - Linux/iOS: https://github.com/nvm-sh/nvm
     - Windows: https://github.com/coreybutler/nvm-windows

NPM is the package manager for NodeJS (similar to pip in Python) and is installed automatically with NodeJS. With NPM we can install all necessary external libraries we need. All installed libraries of a project are listed in a `package.json`-file. Installed dependencies are stored in the `node_modules/`-folder.

2. Import third-party libraries
   - Importing external libraries is done similarly as internal files. But to track all used libraries and their version numbers, we initialize the `package.json`-file beforehand.
   - Run the following command and go through the initialization process (just pressing **enter** everytime is sufficient):

     ```
     > npm init
     ```

   - Now install the needed dependencies with the following command:

     ```
     > npm install
     ```

Note: If you plan to share your code with `git` or a cloud sharing service (e.g. `nextcloud, google drive, onedrive, dropbox, ...`), you should **not** share the `node_modules/`-folder. Add `node_modules/` to your `.gitignore`-file or equivalent. Everyone can install all dependencies listed in the `package.json`-file automatically by executing the following command:

```
> npm install
```

Task 1.2 Starting the Webserver

The package.json file contains the following scripts, that can be executed on the command line:

1. Starting the development server. Saving a file will immediately update the webpage as well.

   ```
   npm run dev
   ```

   this starts the webserver. In your browser you can type `localhost:3000` which allows you to visit the page. Pressing the Load Data button should show you the image shown in Figure 1.

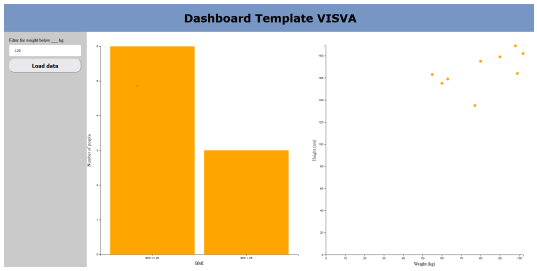

Abbildung 1: If everything went correctly, you should see this image after starting the webserver.

2. Optimizing the code for deployment. Will often have significant performance improvements, but harder to debug.

   ```
   npm run build
   ```

3. Start the webserver for the transpiled code

   ```
   npm run server
   ```

Task 1.3 Examples

There are a number of examples in the code to show you various aspect of how to work with a webserver and front-end, as well as how to use d3.

- **src/_server/Static/server.js** contains an example how to create a webserver with nodejs and express
- **src/_public/index.js** contains an example how the webclient can request data from the server.
- **src/_server/websocket.js** contains an example how to send data from the server to the client upon a request.

- **src/_server/websocket.js** contains an example how to read data from a file on the server.
- **src/_server/preprocessing.js** contains an example how to preprocess data on the server.
- **src/_server/websocket.js** contains an example how to filter the data on the server.

- **src/html/template.html** and **src_/public/app.css** contain an example how to layout a dashboard.
- **src/html/template.html** and **src_/public/index.js** contain an example how to include buttons and input fields for parameterized data requests.
- **src/_public/src/barchart.js** and **src/_public/src/scatterplot.js** contain two example visualizations that use the data.

We would highly recommend that you read through the code of each example, and try to understand what it is doing before starting the project itself. During the remainder of the projects, you will be using, adapting and creating new code based on this.

Task 1.4 Deliverables

There will be no deliverables for Task 1. However, completing this task will help you with all projects.

Task 2 Visualizing a dataset

In Task 2 you will use the template from Task 1 to design a visualization to analyze a dataset from start to finish. As you go through the tasks, document your process, in particular which decisions were taken and *why* these decisions were taken. This includes but is not limited to choosing analysis tasks, how the data is preprocessed, which visualizations to use, and so on.

All libraries/dependencies in the template are allowed. Additional dependencies are not allowed unless permission is given by your instructor.

Ensure that the work you perform is your own work. You may take inspiration from code and designs from outside sources, but copy-pasta is not allowed. When using code as inspiration, ensure that you explain which changes you have made to make it suitable for your own purposes in both comments in the code and the presentation. When in doubt if the changes to the design are significant enough, ask your instructor.

For this Task, you will be using data from the Graph Drawing 2023 Contest (https://mozart.diei.unipg.it/gdcontest/2023/creative/), which collected a dataset from BoardGameGeek about board games. The provided data (boardgames_40.json) contains a cleaned dataset with information about the top-40 boardgames and is in .json format. A description of all fields is given below:

- id: An unique id with which the boardgame can be identified.
- Year: Contains the year the game was published.
- minplayers: Contains the minimum amount of players required to play the game.
- maxplayers: Contains the minimum amount of players required to play the game.
- minplaytime: How long the game is estimated to take at minimum.
- maxplaytime: How long the game is estimated to take at minimum.
- minage: The recommended minimum age for the game.
- rating: Contains a nested json object with the average rating of the game (*rating*) and the number of reviews (*num_of_reviews*)
- Recommendations: Contains a list of *id*'s of other board games that are often liked by people who have liked this game. This list is calculated using an internal algorithm from boardgamegeek.
- types.categories: Contains id,name pairs of what type of game this falls under. Id's are unique for the categories.
- types.mechanics: Contains id,name pairs for what kind of mechanics (Is it a cooperative? Are there teams? Can you trade with other players?, etc.)
- credit.designer: Contains id,name pairs for who made this game

For this first project, you will only be needing a subset of the data, in particular the *recommendations* object should not be used yet.

Task 2.1: Preceding Visualization: Exploration

Derive aspects of the data that could be of interest to a board-game developer when exploring the dataset, and formulate one analysis task that an analyst might want to perform on the data with the Group characteristic. Specify the task using the 5-tuple (goal (Explore), means, characteristics (Group), target, cardinality) as presented in the lecture.

After having decided upon the analysis task, decide on a preceding visualization you want to use to support this task. Note that simply using the bar chart or scatterplot provided in the template as is, is **not** sufficient.

Using the knowledge from the lecture, argue why this visualization and the design choices made are appropriate for your task 5-tuple.

Once you have decided on the visualizations to use, determine how you will pre-process the data. After preprocessing and verifying that this works correctly, implement your visualizations using the template D3 to visualize your dataset. Finally, add basic elements such as labels, grid lines, colors, etc., to the visualizations when they help in addressing the analysis task or prevent confusion.

Task 2.2 Subsequent Visualization: Dimensionality Reduction via LDA

In this task, you will analyze the data using the LDA dimensionality reduction technique. You can use the dimensionality reduction library from https://www.npmjs.com/package/@saehrimnir/druidjs for the algorithm itself.

You can install this package and all required dependencies via:

```
npm install @saehrimnir/druidjs
```

Before you apply LDA to the data, take an initial look at the data and come up with an analysis task (5 tuple) that could be interesting for an analyst for a Compare action, and that can be supported by LDA in particular. Document the analysis task and explain why LDA is appropriate for this.

You will then work with the *model visualization* component of the VA pipeline. visualize the outcome of the LDA model for this specific analysis task. Preprocess the data in a suitable way, and design and implement a visualization that supports this analysis task that uses LDA to visualize the data for comparison. Be sure to write down why this particular Subsequent visualization is suitable.

Task 2.3: Tightly integrated visualization: Interacting with LDA paramaters

In this task, you will work on the *Model Building* component of the VA pipeline. In particular, you will work with interacting with the parameters of LDA, changing either the amount of output dimensions, or the classes chosen. When designing the Visualization for Task 2.2, take these constraints into account. Write down how you can use the implemented interaction to analyze your data, and implement it.

Important note: be sure to schedule time for the project to allow for people to work on interactions after the other visualizations are finished. While interactions tasks can start before the visualizations are done, they need to be nearly ready before interaction tasks can be finished.

Deliverables

There are two deliverables for this project, a working version of the code and a presentation. Both have to be uploaded to Ilias **before 18:00 23.05.2025**. Please don't upload the `node_modules`-folder and its content.

Code

A working version of the code should be uploaded to Ilias. This includes all files used to generate the visualizations, as well as any files that have been used to preprocess the data.

Presentation

During the tutorial in the week of 27.05.2025, a presentation session will be held. Each group will present their work in at most 10 minutes, with an additional 3 minutes for questions. A single individual from each group should present their work, while the remainder of the group should be ready to address the questions. Time will be strict, so preparation and a trial run is recommended. All individuals should be present during the presentation.

The presentation should consist of a small live demo of your implemented visual design, and a PowerPoint (or similar) presentation going explaining the design and why design decisions were made. At the end of the presentation, questions will be asked to individual members of the group for further clarification and to test understanding of the proposed design, see below for a list of example questions.

Suggested outline of the presentation with indications of how much time spend:

1. Intro slide (0.5 min)
2. Here is the analysis task for 2.1 and 2.2 that we came up with. These could be interesting to board game developers for the following reasons. (1 min)
3. This is the implemented visualization for Task 2.1, these are the techniques used. (1 min).
4. Here's how it supports the chosen analysis task well using terminology from the lecture(0.5 min)
5. This is the implemented visualization for Task 2.2, these are the techniques used. (1 min).
6. Here's how it supports the chosen analysis task well using terminology from the lecture (0.5 min)
7. Live demo demonstrating the visualization. Show how to use the interaction. (2 min)
8. Short explanation of what could be improved (0.5 min)

Feel free to deviate from this outline as necessary, but make sure to cover the points mentioned in the grading scheme.

Presentation first slide constraints

The first slide of your presentation should have at least the following details:

- The names of the people in the group.
- The analysis tasks that you came up with.

- The distribution of the work over the members of the group.
- The rough time spent on each part of the project.
- A screenshot of the final visual design.

When uploading the presentation, ensure that it is in a .ppt, .pptx, or a .pdf file format.

Example questions

Below, we present examples of the type of questions that you may be asked during the presentation.

1. When setting up the environment, was there anything that confused you at first?
2. What did you initialize notice in the data when exploring it?
3. Why did you preprocess the data in this way?
4. How have you managed this particularity in the data?
5. Can you explain why you have chosen this analysis task to explore?
6. How is analysis task supported?
7. Why have you chosen for this type of visualization for this analysis task?
8. Why is this basic element (not) included.
9. Did you find any interesting insights using your visualization?
10. What changes did you make to the original visual design?
11. What was the most difficult part to design?
12. Given more time, what would you add?
13. If you did the exercise again, what would you have done differently?
14. What would you have liked to incorporate into your design from the other presentations seen so far?
15. Are there questions, that can not be asked with your visualizations, but can be answered with the data itself?

Grading scheme

Below, we present the list of criteria that the group will be graded on.

- ☐ A 5 tuple-task exploring the data is present using the Group characteristic (1pt), and the reasoning why it could be interesting for a board-game developer is clear and sound (1pt) (Task 2.1)
- ☐ A 5 tuple-task for LDA is present using the Compare action (1pt), and the reasoning why it could be interesting for a board-game developer is clear and sound (1pt) (Task 2.2)
- ☐ A visualization has been implemented for Task 21. (2pt)
- ☐ Argumentation for why the implemented Visualization for Task 2.1 support the chosen analysis task is clear and sound. (1pt)
- ☐ A visualization has been implemented that uses LDA for Task 2.2. (2pt)
- ☐ Argumentation for why the implemented Visualization for Task 2.2 support the chosen analysis task is clear and sound. (1pt)

☐ Argumentation for why basic elements are included and excluded in both implementation is clear and sound. (1pt) (Task 2.1 and 2.2)

☐ An interaction with the parameters of LDA has been implemented and its use shown. (2pt) (Task 2.3)

☐ The cost of interaction has been kept as low as possible. (2pt) (Task 2.3)

☐ There is a reflection on the potential improvements of the implemented VA system and/or the process of developing it. (1pt)

☐ The live demo during the presentation worked completely and matched the designs presented. (1pt)

☐ Most questions after the presentation were answered to satisfaction. (2pt)

☐ All questions after the presentation were answered to satisfaction. (1pt)