TP2 (Python Version)

INF8808 : Data Visualisation

Department of computer and software engineering

Author : Olivia Gélinas

Teacher's Assistant : [Hellen Vasques]

Objectives

The goal of this lab is to create an interactive stacked bar chart using open data in CSV format.

Before beginning, we recommend you have completed the following readings and practice exercises :

Readings :

- Bar Charts

  > https://plotly.com/python/bar-charts/

- Theming and Templates

  > https://plotly.com/python/templates/

- Hover Text and Formatting

  > https://plotly.com/python/hover-text-and-formatting/

- Basic Callbacks

  > https://dash.plotly.com/basic-callbacks

Exercices : TP2 exercices : 1, 2, 3, 4

Introduction

A bar chart is a type of data visualisation which represents data using rectangles whose lengths are proportional to the values they represent. A stacked bar chart extends the basic bar chart by allowing us to group data into a first categorical variable, and using colors to divide each bar into a second-level categorical

variable. A stacked bar chart is useful when we are interested in the total size of each group as well as the size of each subgroup.

In this lab, you will implement a stacked bar chart using data from Shakespeare's play, *Romeo and Juliet*. The data was extracted from Kaggle, a popular data science website [1]. The dataset was then modified into a version to be used in this lab. It contains data on the entire text of *Romeo and Juliet*.

Description

In this lab, you will have to complete the Python code using Plotly and Dash in order to display a stacked bar chart displaying data on how many lines each player has in each act of the play.

To make the chart interactive, you will implement the code to toggle between two display modes : "Count" and "Percent". In the "Count" mode, the chart displays the line count for each player directly. In "Percent" mode, the line count for each player is relative and represents the proportion of lines a given player has in a given act. Also, each stack will contain as subgroups one bar for each of the top 5 players who utter the most lines in the play. The other players will be grouped together into their own category "Others", containing their summed line count and percentage.

To customize the appearance of the chart, you will also implement the code to create a new theme as well as a template for a tooltip that appears when each bar is hovered by the mouse.

The following subsections present the different parts that you will have to complete for this lab. While you code, we recommend completing the data processing first, followed by the implementation of the bar chart itself. The creation of the theme and the tooltip template are independent from the other parts. The toggle between the two display modes may be implemented last.

File Structure

To complete this work, you will need to fill the various "TODO" sections in the files from the archive provided for the lab. The comments in the code explain in more detail the steps to take. The scripts to use are located in the "assets" directory of the archive provided for the lab.

In this lab, we provide you with and archive with 7 Python files used to accomplish the desired visualization :

- `app.py` : This file generates the HTML structure of the webpage and orchestrates the steps required to create the visualization.
- `bar_chart.py`
- `hover_template.py`
- `modes.py` : This file contains some constants that may be used to help manage the two display modes. You do not need to modify it.
- `preprocess.py`
- `server.py` : This file is used to launch the application. You do not need to modify it.
- `template.py`

Dataset

The dataset is located in the `src/assets/data/` directory in the archive provided for the lab. The dataset contains the following columns :

- `Act` : This column represents the act in which the line was uttered.
- `Scene` : This column represents the scene in which the line was uttered.
- `Line` : This column tracks the number of the line for the given act and scene.
- `Player` : This column contains the name of the player who uttered the line.
- `PlayerLine` : This column contains the content uttered by the given player during the given act and scene at the line with the given number.

Note : As a reminder, the dataset represents data from a play, which are usually divided into large chunks called "acts". In this play, there are 5 acts. In turn, each act is divided into scenes, which may contain any number of lines, presented in the order in which they are uttered during the play. Also note that by "player" we are referring to a character played by an actor.

Data preprocessing

To begin, you will have to preprocess the data we provide you. The data contained in the CSV file is raw, so it is necessary to reorganize certain parts of it so they can be properly used by the Plotly library. To do so, you need to complete the file `preprocess.py`.

More precisely, you will have to complete these steps :

1. For each act, group each player's lines together, summing each player's line count for the act and including the corresponding percentage of lines for that player in that act (function `summarize_lines`)
2. Modify the structure to include an "Other" category, which contains the sum of the count and percentage of lines in that act of players not belonging to the top 5 for the play (function `replace_others`)
3. Update the names of the players in the dataset so they follow correct capitalization (function `clean_names`)

Figure 1 illustrates a portion of the resulting dataset to help validate your work.

Act	Player	LineCount	LinePercent
1	Benvolio	34	14.406780
1	Juliet	16	6.779661
1	Mercutio	11	4.661017
1	Nurse	20	8.474576
1	Other	105	44.491525
1	Romeo	50	21.186441

Figure 1 : Extract of preprocessed data

Bar chart

For this second part, you will have to implement the main part of the data visualization. First, you will have to display the appropriate data in the bar chart, depending on the selected display mode. Second, you will adjust the y-axis label depending on the selected display mode. See the comments in the code for more details on the text to display. To complete this part, you will need to modify the file `bar_chart.py`.

These are the steps you will have to accomplish for this part :

1. Complete the definition of the chart's figure to include the template and title to be used (function `init_figure`)
2. Display the appropriate data in the bar chart depending on the current mode (function `draw`)
3. Display the appropriate text on the y-axis depending on the current mode (function `update_y_axis`)

To see the result of this part, partially completing the `radio_updated` function in `app.py`, used to toggle between display modes, may be useful.

In Figure 2 and in Figure 3 we can see what the resulting bar chart should look like in both "Count" and "Percent" display modes, according to which portion of the preprocessed data is displayed.

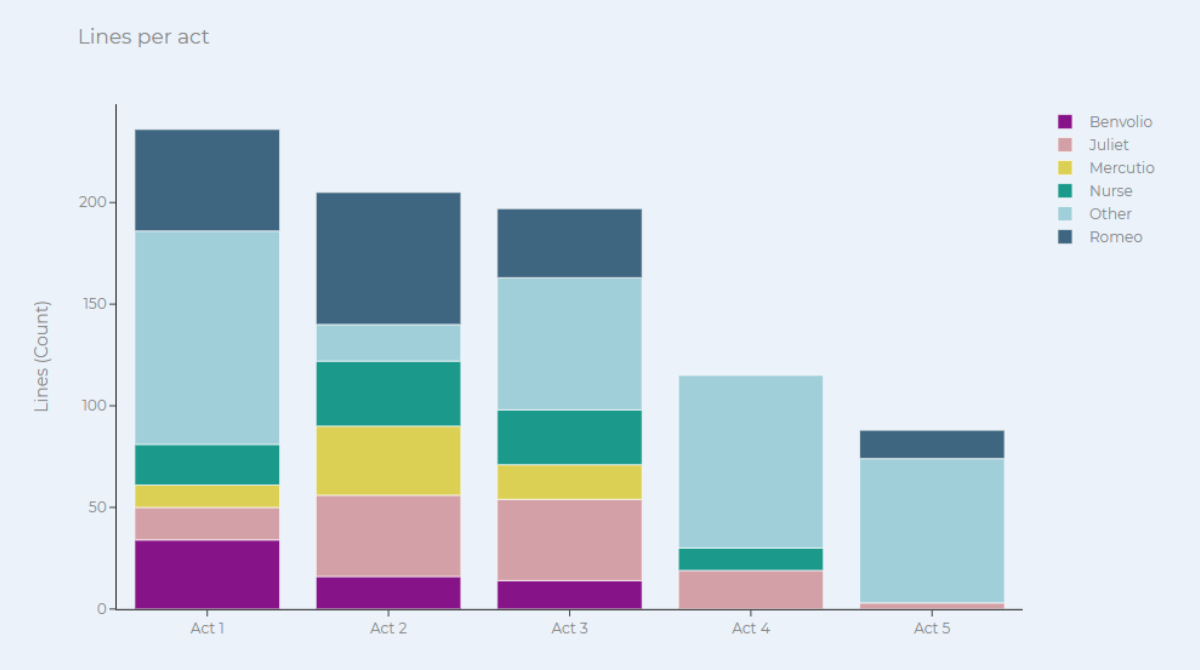

Figure 2 : The Stacked bar chart in display mode "Count"

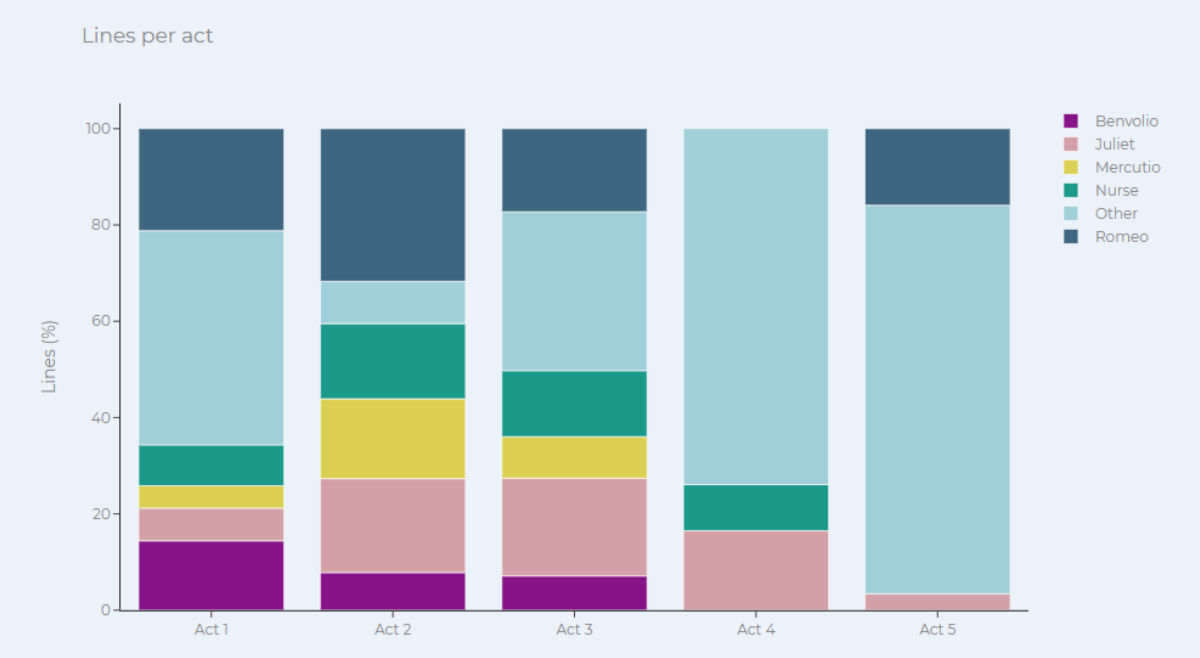

Figure 3 : The Stacked bar chart in display mode "Percent"

Theme

For this third part, you will create a custom template to customize the visual display of the bar chart. The code for this part is in the file `template.py`. You need to complete the function `create_template`. Make sure to closely follow the instructions in the comments of the code when defining each element of the theme. The desired values to use are defined in a variable at the beginning of the file. Make sure the colors in the bar chart appear in the same order as in Figure 2 and Figure 3 which also illustrate the theme's result.

Tooltip

For this fourth part, you will define a template used to display the tooltip that appears when the mouse hovers over a bar. The tooltip should contain as a title the act associated to the bar's group, as well as the bar's player and line count or percentage (depending on the current mode). The comments in the code give a detailed description of the contents and appearance of the tooltip. See Figure 4 for the expected final result. The code for this part should be written in the file `hover_template.py` in the function `get_hover_template`.

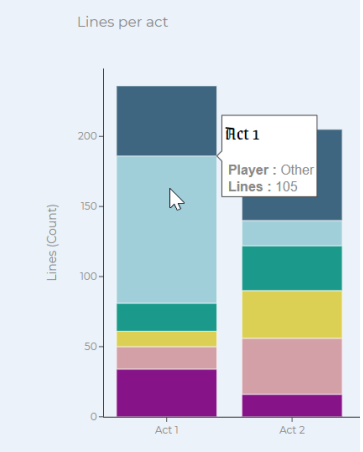

Figure 4 : The Stacked bar chart with tooltip in display mode "Count"

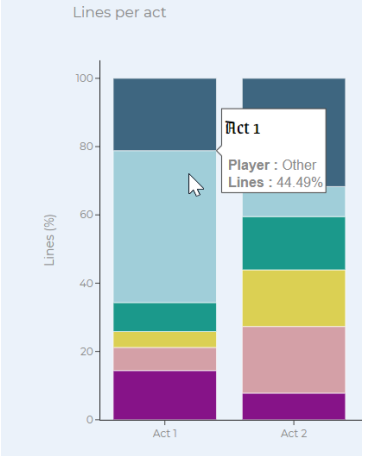

Figure 5 : The Stacked bar chart with tooltip in display mode "Percent"

Display mode toggle

For this fifth part, you will implement a method to toggle between the chart's to display modes based on the radio button input. Whenever the input on the radio button is modified, the bar chart should be modified correspondingly. The displayed data should be selected from the appropriate column in the dataset depending on the selected mode. The tooltip and y-axis should also display texts corresponding to the selected mode. Further, the information text at the left of the footer should contain the currently selected mode. Figure 5 gives an example of the footer's appearance when the selected display mode is "Percent".

To complete this part, you will need to find the appropriate functions to call in the `radio_updated` function in `app.py`. Beginning by partially completing this function may also be helpful for visualizing your bar chart while you complete the previous steps.

Figure 6 : The information displayed in the foot when the display mode is "Percent"

Submission

The instructions for the submission are :

1. You must place your project code in a compressed ZIP file named [matricule1_matricule2_matricule3.zip]

2. The lab must be submitted before [May 22nd 23h59]

Evaluation

Overall, your work will be evaluated according to the following grid. Each section will be evaluated on correctness and quality of the work.

Requirement	**Points**
Data preprocessing	6

Requirement	Points
Bar Chart	7
Theme	2
Tooltip	2
Display mode toggle	2
Overall quality and clarity of the submission	1
Total	**20**

References

[1] L. Larsen, "Shakespeare plays," Kaggle, [Online]. Available: https://www.kaggle.com/kingburrito666/shakespeare-plays. [Accessed 30 07 2020].