DOCUMENTATION ABOUT PYTHON AND R WORKGROUP – Task 2 - Part 1

API: STUDIO GHIBLI

C. Di Razza, M. Gioia, V. J. Striano

First, we strongly suggest reading the first task's documentation.

After the data collection and cleaning task, we performed some data analysis based on graphs.

We used one main library *matplotlib* to create graphs. Its documentation is available at:
https://matplotlib.org/stable/gallery/index.html

GridSpec from matplotlib was used to create some advanced subplots. A subplot is an image composed of multiple plots with a fixed size. GridSpec creates subplots with different proportions arrangements. GridSpec's documentation is available here:
https://matplotlib.org/stable/api/_as_gen/matplotlib.gridspec.GridSpec.html

Numpy is broadly used to create smarter numerical lists. Numpy arrays take a third of the bit needed to storage lists, making them way lighter. In our case, numpy is involved because matplotlib uses its arrays to set the tickers for plots. Its documentation available here:
https://numpy.org/doc/stable/reference/generated/numpy.array.html

.plot function from *Pandas* create matplotlib graphs, in a one-line command, let us save a huge amount of lines. Unfortunately, that doesn't make plots customizable. Pandas' documentation is available in the first document.

For each dataset, we created one gridspec plot with multiple graphs. For the same type of plots, we coded different approaches (either with Pandas one-code line and Matplotlib).

[4] import numpy as np

[5] from matplotlib import pyplot as plt

[6] from matplotlib import gridspec

We import the libraries needed.

As said before, we create different plots for the same data frame within the same image thanks to gridspec. For each gridspec we create a figure with a 16:9 size proportion, to fit modern screens.

[34] fig1 = plt.figure(1, figsize=(16,9))

Then we set the main title for the whole image.

[62] plt.suptitle("Graphical Analysis for the \"Films\" Studio Ghibli API", size=16, y=1)

We use the “ggplot” color palette. Styles available here: https://matplotlib.org/stable/gallery/style_sheets/style_sheets_reference.html

[63] plt.style.use("ggplot")

With gridspec we create a “virtual” grid, which slices the whole image in X rows and Y columns.

[35] gs1 = gridspec.GridSpec(1,2)

Thanks to gridspec.update we set the vertical and horizontal spacings between the plots.

[61] gs2.update(wspace=0.5, hspace=100)

For each plot, we create a subplot that fill the grid in the X rows and Y columns.

[48] subplot1 = fig1.add_subplot(gs1[0,0])

Each subplot has peculiar features. For brevity’s sake, we’ll give explanations for a couple of them.

Regarding the *film’s* dataframe, we pie plot the distribution of genders among characters. In this case, we use the standard matplotlib command. We divided the genders in three main categories: “Male”, “Female” and “Others”. We use .loc Pandas function to filter observation whose variable “gender” is equal to the string “Male” and “Female”.
The .count()[0] argument let us show the number of the filtered observation. To enumerate the “Others” we calculate the number of observations thanks to len(dataframe) and subtract the male and female numbers.

The plt.pie matplotlib function creates a pie plot. The variables involved are “male”, “female”, “others”; labels shown have the same name. Autopct shows the percentages of the variables.

[50] plt.pie([male,female,other], labels = labels, autopct=”%.2f %%”)

The plt.title matplot lib function sets the title for the subplot.

[51] plt.title(“Gender distribution among people”)

Later, we boxplot the statistical distribution for the running time. We set the title and “\n” to increase the spacing between the title and the graph. We set the color, too, according to the ggplot palette.

[79] d["films"]["running_time"].plot(kind='box', title='Statistical distribution for running times\n', color="#0080b0")

We showed the y-axis label named “Minutes”

[80] plt.ylabel(“Minutes”)

Thanks to “tick_params” we hid the x-axis ticks and labels. They are not needed when it comes to showing a boxplot.

[81] subplot3.tick_params(labelbottom=False)

[82] subplot3.tick_params(bottom=False)

Sometimes, the axis labels were a mess! We set the ticks thanks to the command yticks. Its argument was a numpy array from 1 to 15, steps:3. The “rotation” argument is used to avoid overlapping among labels.

[74] plt.yticks(np.arange(1,15,3))

[75] plt.xticks(rotation=5)

The plt.legend() shows the legend for a plot.

[88] plt.legend()

The plt.savefig matplotlib command saves the plot into a .png file. We set a high resolution for better visualization.