07-Exercises_convert

December 4, 2024

1 07 - Exercises: data visualization and numerical data analysis

This week we saw: - What is dataviz and how we can visualize data with Matplotlib - What is numerical data analysis and how we can do it with NumPy

Here are some exercises to help you get comfortable with these concepts :)

1.1 1. Get comfortable with NumPy arrays - Difficulty level: 1 (3 points)

Basic exercise to get your hands dirty with NumPy arrays. 1. Create a NumPy array of integers from 1 to 20. 2. Reshape it into a 4x5 matrix. 3. Perform the following operations: - Calculate the mean, sum, and standard deviation of the array. - Extract the second row and the third column. - Replace all even numbers in the array with 0.

Example output:

```
Original array:
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]

Reshaped array:
[[ 1  2  3  4  5]
[ 6  7  8  9 10]
[11 12 13 14 15]
[16 17 18 19 20]]

Array sum:     210
Array mean:    10.5
Array st. dev.:    5.766281297335398

Second row of reshaped array: [ 6  7  8  9 10]

Third column of reshaped array: [ 3  8 13 18]

Modified array:
[[ 1  0  3  0  5]
[ 0  7  0  9  0]
[11  0 13  0 15]
[ 0 17  0 19  0]]
```

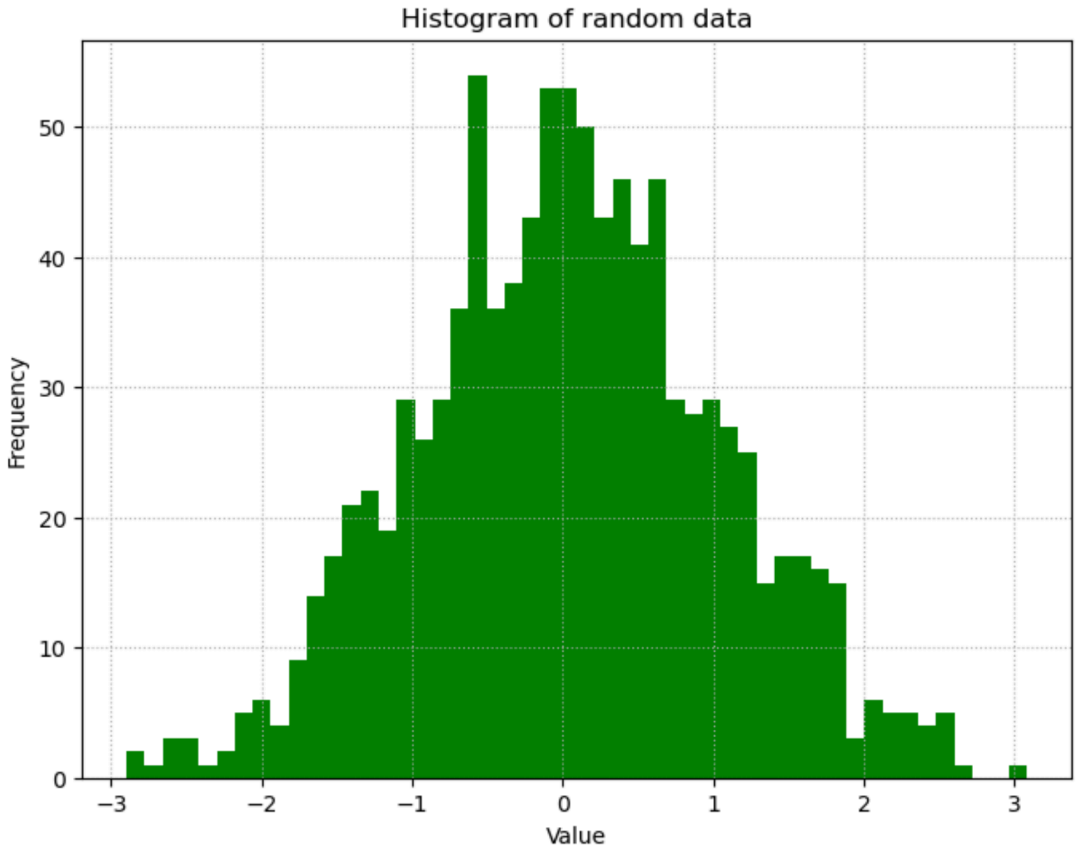
[ ]:

1.2 2. Get comfortable with Matplotlib - Difficulty level: 1 (3 points)

Basic exercise to get your hands dirty with Matplotlib.

1. Generate 1000 random numbers from a standard normal distribution using `np.random.randn`.
2. Plot a histogram of the numbers using Matplotlib.
3. Add the following to your plot:
   - Title: "Histogram of Random Data"
   - Labels for the x-axis and y-axis.
   - Use 20 bins for the histogram and change its color to green.

Example output:

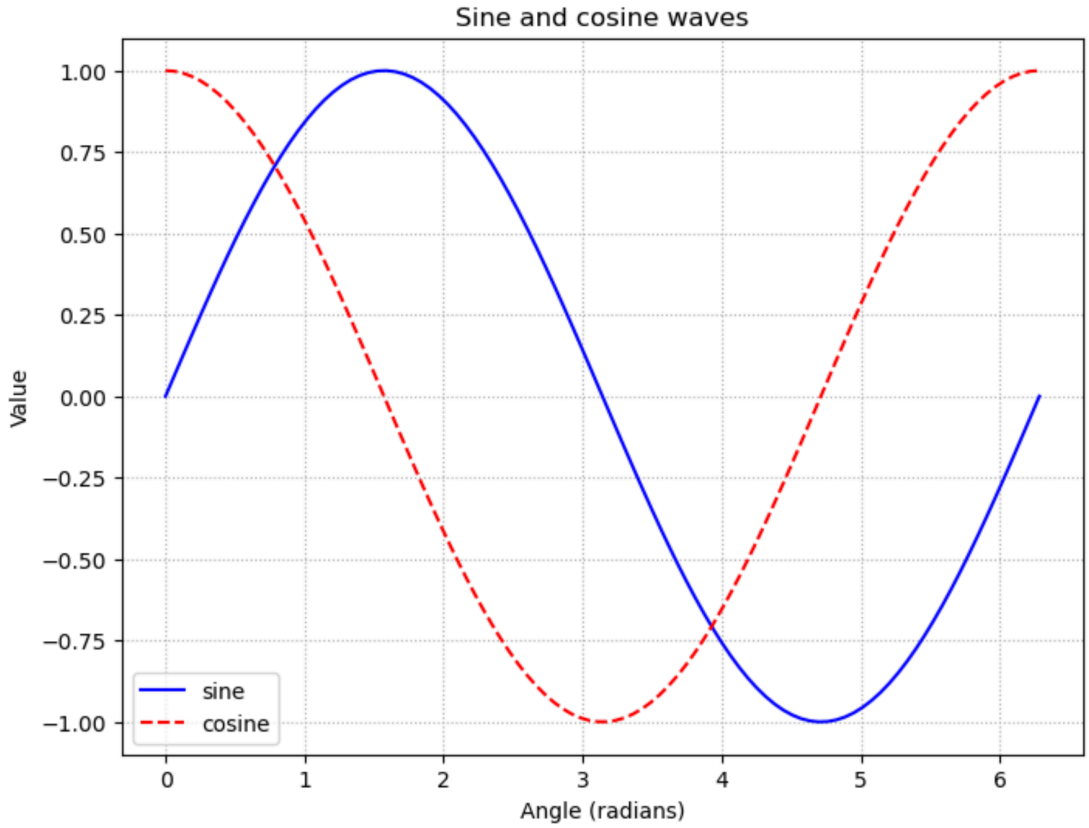


[ ]:

1.3 3. Get comfortable with NumPy Matplotlib - Difficulty level: 2 (3 points)

A less basic exercise to combine NumPy and Matplotlib.

1. Create an array of 100 equally spaced values between 0 and 2π using a NumPy function.
2. Compute the sine and cosine of these values using NumPy functions.
3. Plot both sine and cosine waves on the same graph:
   - Use a solid line for sine and a dashed line for cosine, with different colors.
   - Add a legend to distinguish between the two lines.
   - Set the x-axis label to "Angle (radians)" and the y-axis label to "Value".
   - Add a grid
   - Set a plot title



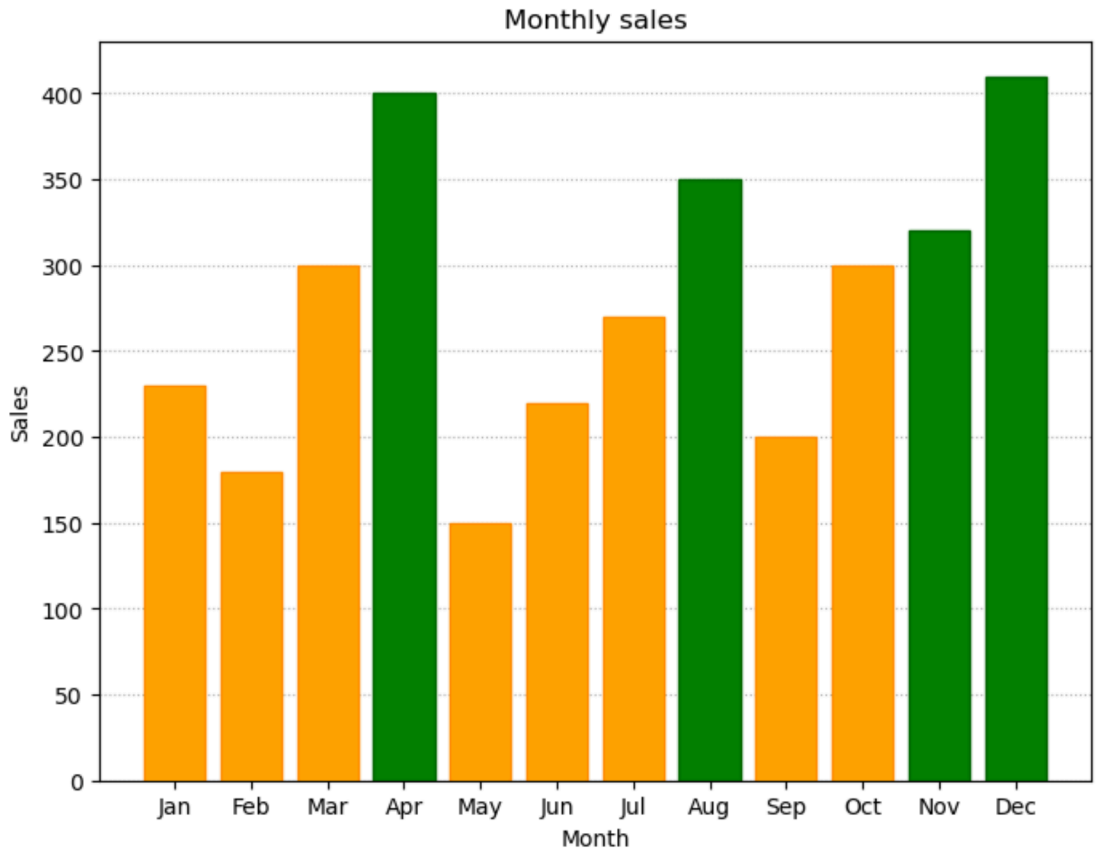
Example output:

[ ]:

1.4 4. More on NumPy Matplotlib - Difficulty level: 2 (3 points)

Another less basic exercise to combine NumPy and Matplotlib.

1. Create a NumPy array with sales data for 12 months: [230, 180, 300, 400, 150, 220, 270, 350, 200, 300, 320, 410].
2. Create a **bar chart** using Matplotlib where:
   - The x-axis represents the months (January to December).
   - The y-axis represents sales.
3. Add the following to your plot:
   - A title: "Monthly Sales"
   - Labels for the x-axis ("Month") and y-axis ("Sales").
   - Color the bars differently for months with sales greater than 300 (use one color for these months and another color for the rest).
   - Add a horizontal grid (Hint: use `zorder` parameter for pushing the grid behind the bars)



Example output:

[ ]:

1.5 5. Our first real data science analysis! - Difficulty level: 2.5 (3 points)

In the lecture, a preprocessed table of climate data from the GISTEMP website (https://data.giss.nasa.gov/gistemp/) has been used. In this exercise, you will perform the preprocessing yourself, using the original table that you can download from https://data.giss.nasa.gov/gistemp/tabledata_v4/GLB.Ts+dSST.csv.

Here are the first lines of the original file, which corresponds to a table stored in comma-separated-values (CSV) format:

```
Land-Ocean: Global Means Year,Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec,J-D,D-N,DJF,MAM,J
1880,-.17,-.24,-.09,-.16,-.09,-.20,-.17,-.09,-.14,-.23,-.21,-.17,-.16,***,***,-.11,-.15,-.19
1881,-.19,-.13,.04,.06,.07,-.18,.01,-.02,-.14,-.21,-.18,-.06,-.08,-.09,-.16,.05,-.06,-.18
1882,.17,.15,.05,-.16,-.14,-.22,-.15,-.06,-.14,-.23,-.15,-.35,-.10,-.08,.09,-.08,-.14,-.17
1883,-.28,-.36,-.12,-.17,-.17,-.07,-.06,-.13,-.20,-.10,-.22,-.10,-.16,-.19,-.33,-.15,-.08,-.18
1884,-.12,-.07,-.35,-.39,-.34,-.35,-.31,-.26,-.26,-.24,-.32,-.30,-.28,-.26,-.10,-.36,-.31,-.28
1885,-.58,-.32,-.25,-.41,-.44,-.42,-.32,-.29,-.27,-.22,-.22,-.09,-.32,-.34,-.40,-.37,-.35,-.24
1886,-.42,-.49,-.42,-.27,-.23,-.33,-.17,-.29,-.23,-.26,-.26,-.24,-.30,-.29,-.33,-.31,-.26,-.25
1887,-.70,-.55,-.34,-.33,-.29,-.23,-.24,-.34,-.24,-.34,-.25,-.32,-.35,-.34,-.50,-.32,-.27,-.28
1888,-.33,-.35,-.40,-.19,-.21,-.16,-.09,-.14,-.11,.03,.04,-.03,-.16,-.18,-.33,-.27,-.13,-.01
1889,-.07,.18,.07,.10,.00,-.09,-.07,-.19,-.23,-.25,-.32,-.28,-.10,-.08,.03,.06,-.12,-.27
```

After the first two (header) lines, each line reports (i) the year in which (ii) monthly temperature measurements have been collected, along with (iii) five additional aggregated measurements. Measurements that could not be collected are indicated by `***`.

Write Python code that converts this file into another CSV table where: 1. Years with incomplete measurements are discarded. 2. The table is transposed, meaning that each column corresponds to measurements taken in a certain year. 3. The row representing the column header lists the years. 4. Only the 12 monthly measurements are reported.

Store this table in a separate file and ensure that it can be loaded with numpy's `loadtxt` function. Your file should be identical to file `Temp_global-mean-monthly.csv` provided in the folder Additional material.

Hint: Use Python's csv reader/writer for this task.

[ ]:

1.6 6. A more structured data visualization exercise - Difficulty level: 3 (5 points)

The course material contains file `12111-04-01-4-B_processed.tsv` which contains demographic data of Germany from the most recent census that was carried out in 2011. The original data is available at https://www.regionalstatistik.de/genesis/online/data?operation=statistic&levelindex=0&levelid=1589541151793

The table is organized as follows: - Columns: federal state (in number code), age, population (male), population (female). - Age `100` summarizes the population that is 100 years or older.

Visualize age distribution in this data set with matplotlib using **three different plots**. Decorate all your plots with title, legend, axis labels, or other features that are necessary to interpret the visualization.

The three plots you need to generate are: 1. Histogram showing the total age distribution of both the male and female population. Details on visualizing multiple data sets in one histogram can be found here https://matplotlib.org/3.1.1/gallery/statistics/histogram_multihist.html. 2. Whisker plots of all 16 federal states showing the total age distribution. You can find a mapping between the number codes and the names of the federal states in file `federal_states.tsv`. 3. Pie chart showing the age distribution of the total population as percentages over age ranges (0, 20], (20, 40], (40, 60], (60, 80), and (100, …)

And finally, generate a fourth plot in which you integrate all previous plots into **one figure** using matplotlib's subplot function. Draw all plots next to each other. Alter the figure size if necessary.

[ ]:

1.7 BONUS. Linear regression exercise - 2 points

Using the same temperature data from exercise 5: 1. Repeat the linear regression analysis shown in the tutorial to fit a line to the observed temperature data, but this time, fit the line to data from every **odd** month (January, March, . . .), and then compute the coefficient of determination using **even** months (Febrary, April, . . .). **Hint**: First think about how you can integrate the monthly data. Do **not** use any aggregator such as mean, min max, etc. 2. Write a function that, given:

- Some `X` coordinates for training, - Some `Y0` coordinates for training, - Some `Y1` coordinates for testing, - And a collection of degrees, fits for each degree a polynomial to the `X/Y0` data and returns the degree with the best fit to the `X/Y1` data set as measured by the coefficient of determination `R2`.

Then apply the function to the temperature data for each month, except for December, which you should use as **testing** data set. Use the range de- grees from 1 to 6 in all your 11 comparisons. Finally, visualize the results using a scatter or line plot.

[ ]: