Visualization
Prof. Bernhard Schmitzer, Uni Göttingen, summer term 2024

Problem sheet 1

- *Submission by* ***2024-04-22*** *18:00 via StudIP* ***as a single PDF/ZIP****. Please combine all results into one PDF or archive. If you work in another format (markdown, jupyter notebooks), add a PDF converted version to your submission.*
- *Use Python 3 for the programming tasks as shown in the lecture. If you cannot install Python on your system, the GWDG jupyter server at https://jupyter-cloud.gwdg.de/ might help. Your submission should contain the final images as well as the code that was used to generate them.*
- *Work in groups of up to three. Clearly indicate names and enrollment numbers of all group members at the beginning of the submission.*

Exercise 1.1: tidying a dataset.
Inspect the dataset contained in the file `runtimes.csv` and import it into Python as a pandas dataframe. Note that one variable (the number of threads) is encoded in column names. Bring this dataset into *tidy* form, as described in `https://r4ds.hadley.nz/data-tidy.html` using functions such as `pandas.melt` (and some post-processing with auxiliary functions). Make sure that columns in the resulting dataframe have meaningful names and dtypes.

Exercise 1.2: basic transformations and visualizations.
The file `mpg-data.csv` contains the `mpg` example dataset from the `ggplot2` library (`https://ggplot2.tidyverse.org/reference/mpg.html`) which contains information about the fuel efficiency of various car models.

1. Import the dataset into Python via pandas and briefly specify the dtype of each column (consult the documentation).
2. Split the dataset into different car classes. For each class, perform a linear regression on the dependency of `hwy` on `displ`.
3. Give a scatter plot of `hwy` against `displ`. Make sure that the class of each car can be determined from the plot. Add straight lines showing the regression lines for each class. Make sure that your plot has appropriate axes labels and legends.
4. Group the data by `class` and `year` and compute the median of `hwy` for each group. Present the resulting dataset as a table.

Exercise 1.3: hue rotation.
Import the photo of parrots used in the lecture (available at `https://en.wikipedia.org/wiki/File:BlueAndYellowMacaw_AraArarauna.jpg`) in Python as shown in the lecture. Then for a given angle $\varphi \in [0, 2\pi)$, implement a function that converts the image to HSV space, applies a rotation by angle φ to the hue channel (where $\varphi = 2\pi$ would correspond to a whole rotation and is thus equivalent to $\varphi = 0$), and transforms the resulting image back to RGB space. Apply this function to the image for $\varphi \in \{\frac{k}{2\pi} | k \in \{0, 1, 2, 3, 4\}\}$ and visualize the obtained 'rotated' images.

Exercise 1.4: visualizing the exponential function.
The exponential function takes complex numbers to complex numbers, $\mathbb{C} \ni z \mapsto e^z \in \mathbb{C}$. Identifying $\mathbb{C}$ with $\mathbb{R}^2$ in the usual way, $z = x + i \cdot y$ for $z \in \mathbb{C}$, $(x, y) \in \mathbb{R}^2$ and i being the imaginary unit, the exponential function can be interpreted as a map from $\mathbb{R}^2$ to $\mathbb{R}^2$. In this representation, it can be written as

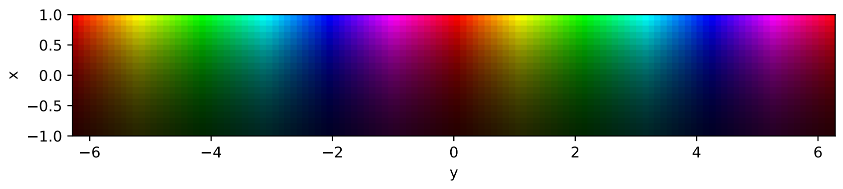
$$\begin{pmatrix} x = \mathrm{Re}(z) \\ y = \mathrm{Im}(z) \end{pmatrix} \mapsto \begin{pmatrix} e^x \cdot \cos(y) \\ e^x \cdot \sin(y) \end{pmatrix}.$$

This exercise aims at visualizing this function by using the HSV color space.

1. Create a Cartesian grid of points over $(x, y) \in [-1, 1] \times [-2\pi, 2\pi]$. Evaluate the exponential function for all points on the grid. Translate the result to polar coordinates (similar as shown in the lecture).

2. As in the lecture, create an empty array for the HSV image over the grid. Set the hue values according to the angle of the polar coordinates, set the saturation to 1, find a suitable map from the radius of the polar coordinates to the value channel. Convert the image to RGB and display the result.

The outcome should look similar to this (rotated for better formatting):


1.0
0.5
x
0.0
−0.5
−1.0
−6
−4
−2
0
2
4
6
y