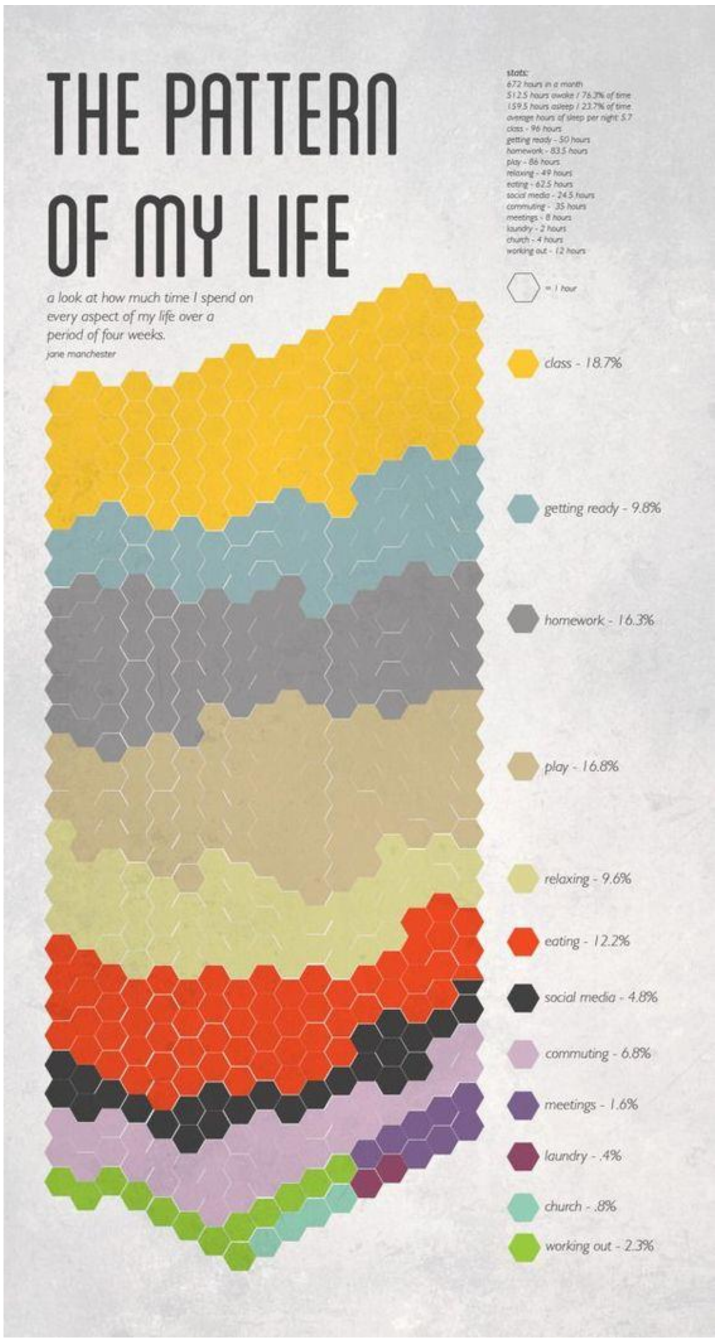
# Data Visualization: Assignment 1

Student 1: Maria Musiał 156062

Student 2: Jakub Liszyński 156060

## Original image



URL: https://pl.pinterest.com/pin/370069294391123816/

Context: Visualization presents day of life of a bored art school student.

Faults:

- False comparison – play segment seems much bigger then homework, however actual difference isn't that big
- The order of categories is confusing and unintuitive – activities that take less time are higher in the list
- Irregularity of graph (hexagons) make it hard to read
- Some hexagons are divided without apparent reason for it

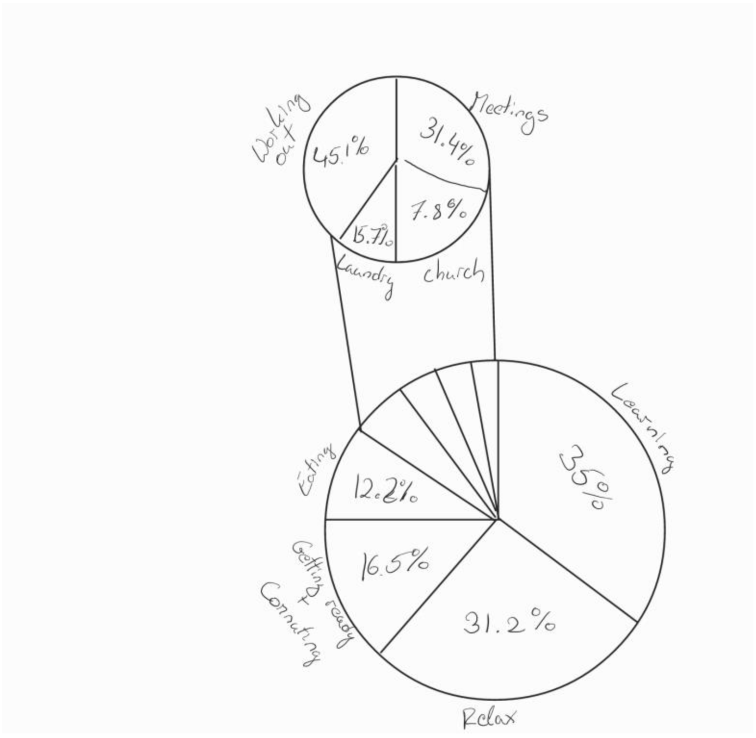
Implications:

- Misreading the information
- Impression that graph is incorrect

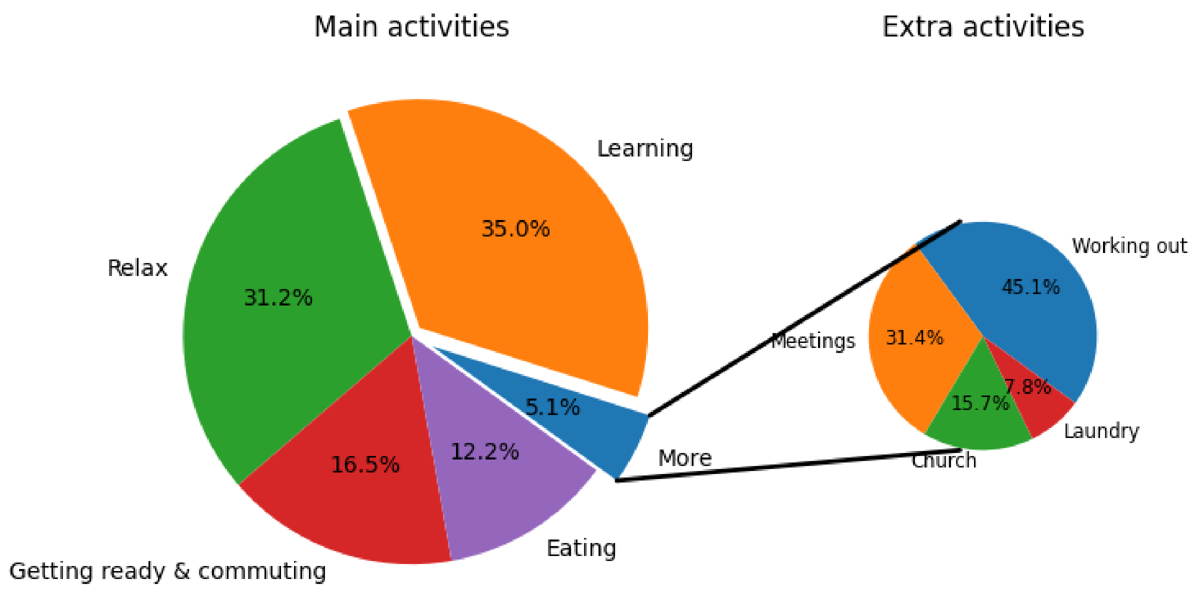
Possible improvements:

- Changing it to a shape that makes it comparable
- Sorting the activities by their percentage
- Stay with pattern; don't split only some of the hexagons

## Sketch



## Implementation



## Implementation code

```python
import matplotlib.pyplot as plt
from matplotlib.patches import ConnectionPatch
import numpy as np

# make figure and assign axis objects
fig = plt.figure(figsize=(9, 5.0625))
ax1 = fig.add_subplot(121)
ax2 = fig.add_subplot(122)
fig.subplots_adjust(wspace=0)

# large pie chart parameters
ratios = [5.1, 35, 31.2, 16.5, 12.2]
labels = ["More", "Learning", "Relax", "Getting ready & commuting", "Eating"]
explode = [0.1, 0.05, 0, 0, 0]
# rotate so that first wedge is split by the x-axis
angle = -180 * ratios[4]
ax1.pie(ratios, autopct='%1.1f%%', startangle=angle,
        labels=labels, explode=explode)

# small pie chart parameters
ratios = [2.3, 1.6, .8, .4]
labels = ["Working out", "Meetings", "Church", "Laundry"]
width = .2

ax2.pie(ratios, autopct='%1.1f%%', startangle=angle,
        labels=labels, radius=0.5, textprops={'size': 'smaller'})

ax1.set_title('Main activities')
ax2.set_title('Extra activities')

# use ConnectionPatch to draw lines between the two plots
# get the wedge data
theta1, theta2 = ax1.patches[0].theta1, ax1.patches[0].theta2
center, r = ax1.patches[0].center, ax1.patches[0].r

# draw top connecting line
x = r * np.cos(np.pi / 180 * theta2) + center[0]
y = np.sin(np.pi / 180 * theta2) + center[1]
con = ConnectionPatch(xyA=(- width / 2, .5), xyB=(x, y),
                      coordsA="data", coordsB="data", axesA=ax2, axesB=ax1)
con.set_color([0, 0, 0])
con.set_linewidth(2)
ax2.add_artist(con)

# draw bottom connecting line
x = r * np.cos(np.pi / 180 * theta1) + center[0]
y = np.sin(np.pi / 180 * theta1) + center[1]
con = ConnectionPatch(xyA=(- width / 2, -.5), xyB=(x, y), coordsA="data",
                      coordsB="data", axesA=ax2, axesB=ax1)
con.set_color([0, 0, 0])
ax2.add_artist(con)
con.set_linewidth(2)

plt.show()
```