matplotlib_workshop

March 2, 2021

1 An introduction to making scientific figures in Python with the Matplotlib visualization library

1.1 Tijs van Lieshout

1.2 2nd of March 2020

[1]:
```
print("An interactive demo/workshop!")
```

```
An interactive demo/workshop!
```

2 Why matplotlib?

2.1 "Matplotlib makes easy things easy and hard things possible."

- Low-level > good for learning / understanding the basics
- Works fine for simple data and python datatypes

3 Installation

[2]:
```
pip install matplotlib
```

```
Requirement already satisfied: matplotlib in
/home/tijs/virtualenvs/programming1/lib/python3.7/site-packages (3.3.3)
Requirement already satisfied: pillow>=6.2.0 in
/home/tijs/virtualenvs/programming1/lib/python3.7/site-packages (from
matplotlib) (8.0.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in
/home/tijs/virtualenvs/programming1/lib/python3.7/site-packages (from
matplotlib) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in
/home/tijs/virtualenvs/programming1/lib/python3.7/site-packages (from
matplotlib) (1.3.1)
Requirement already satisfied: python-dateutil>=2.1 in
/home/tijs/virtualenvs/programming1/lib/python3.7/site-packages (from
matplotlib) (2.8.1)
Requirement already satisfied: numpy>=1.15 in
/home/tijs/virtualenvs/programming1/lib/python3.7/site-packages (from
```

```
matplotlib) (1.19.4)
Requirement already satisfied: cycler>=0.10 in
/home/tijs/virtualenvs/programming1/lib/python3.7/site-packages (from
matplotlib) (0.10.0)
Requirement already satisfied: six in
/home/tijs/virtualenvs/programming1/lib/python3.7/site-packages (from
cycler>=0.10->matplotlib) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
```

[3]:
```
import matplotlib.pyplot as plt
```

4 Basics of matplotlib

4.1 Explained with data from kittens

5 The dataset

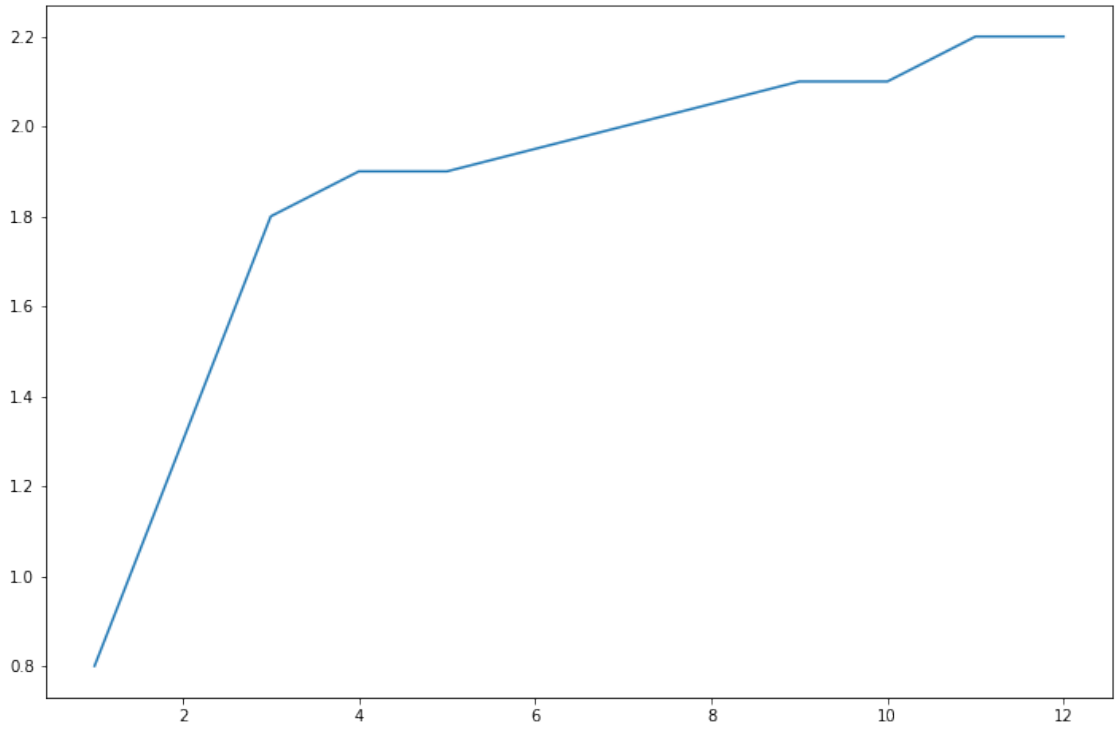
5.1 Cat weight in kg by age in months

[4]:
```
# data from https://www.healthynex.com/cat-weight-chart-by-age-in-kg-ib.html
cat_age_in_months = [1, 2, 3, 4, 5, 7, 9, 10, 11, 12]
cat_weight_in_kg = [0.8, 1.3, 1.8, 1.9, 1.9, 2.0, 2.1, 2.1, 2.2, 2.2]
```

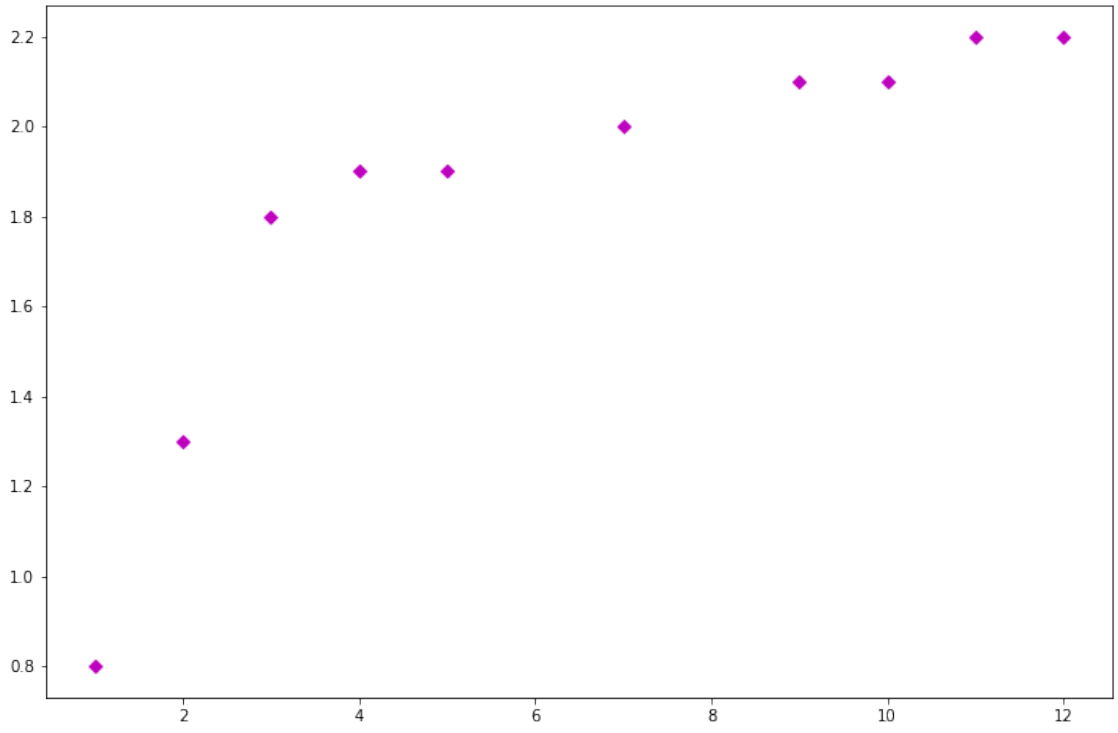
[5]:
```
print(len(cat_age_in_months) == len(cat_weight_in_kg))
```

```
True
```

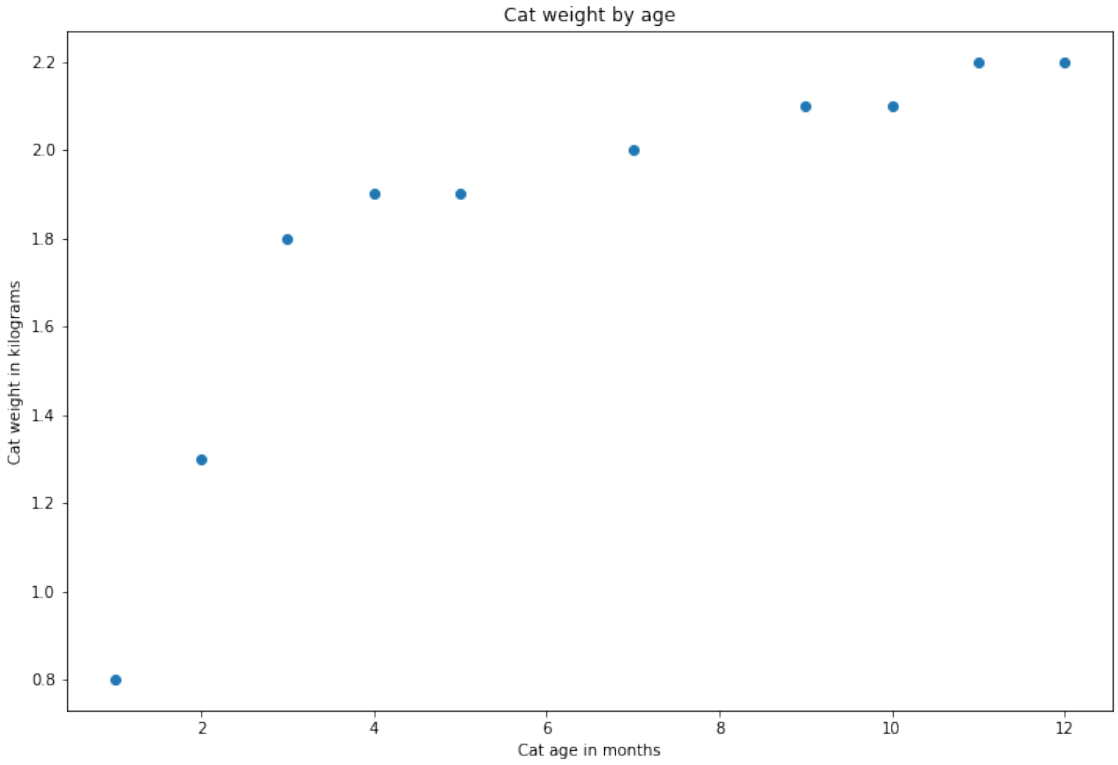
[6]:
```
plt.figure(figsize=(12, 8))
plt.plot(cat_age_in_months, cat_weight_in_kg)
plt.show()
```



[7]:
```
plt.figure(figsize=(12, 8))
# o, ro, k+, c--, mD
plt.plot(cat_age_in_months, cat_weight_in_kg, 'mD')
plt.show()
```

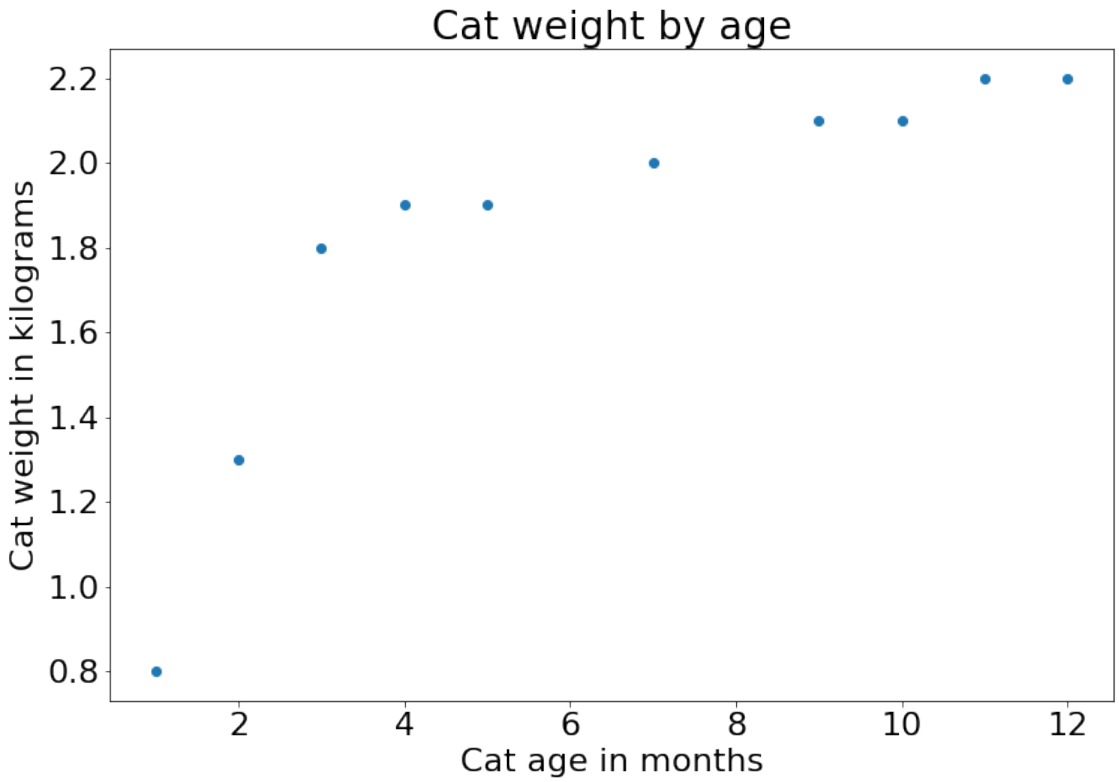


[8]:
```
plt.figure(figsize=(12, 8))
plt.plot(cat_age_in_months, cat_weight_in_kg, 'o')
plt.xlabel('Cat age in months')
plt.ylabel('Cat weight in kilograms')
plt.title("Cat weight by age")
plt.show()
```



[9]:
```
plt.rcParams.update({'font.size': 22})
```

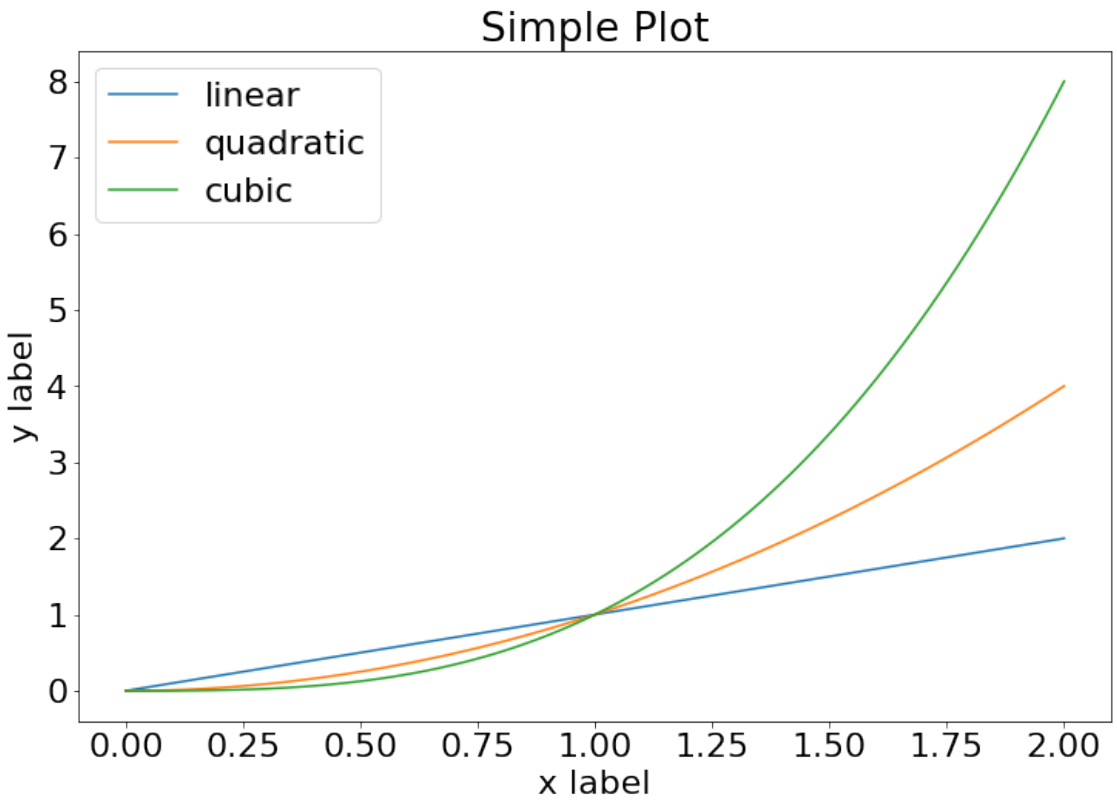
[10]:
```
plt.figure(figsize=(12, 8))
plt.plot(cat_age_in_months, cat_weight_in_kg, 'o')
plt.xlabel('Cat age in months')
plt.ylabel('Cat weight in kilograms')
plt.title("Cat weight by age")
plt.show()
```



6 Pyplot vs. Object-oriented style

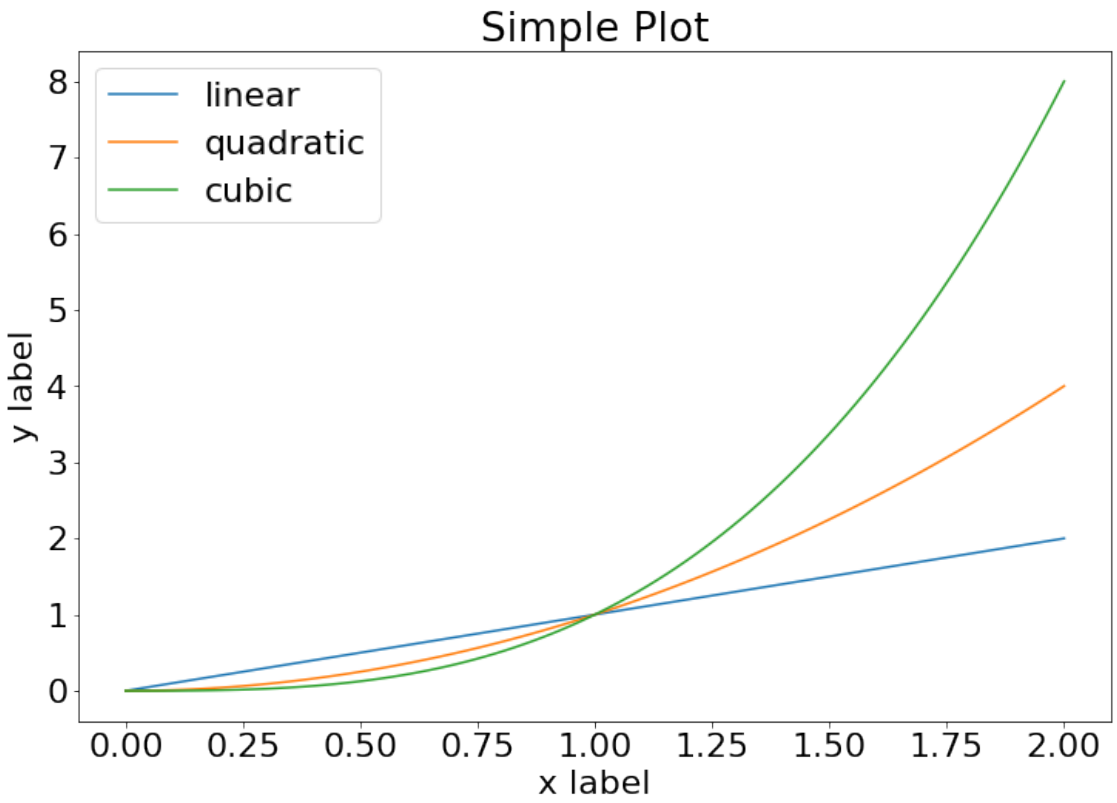
[11]:
```
# adapted from https://matplotlib.org/stable/tutorials/introductory/usage.html
import numpy as np
x = np.linspace(0, 2, 100)

plt.figure(figsize=(12, 8))
plt.plot(x, x, label='linear')  # Plot some data on the (implicit) axes.
plt.plot(x, x**2, label='quadratic')  # etc.
plt.plot(x, x**3, label='cubic')
plt.xlabel('x label')
plt.ylabel('y label')
plt.title("Simple Plot")
plt.legend()
```

[11]:
```
<matplotlib.legend.Legend at 0x7f316df46290>
```



[12]:
```
# adapted from https://matplotlib.org/stable/tutorials/introductory/usage.html
import numpy as np
x = np.linspace(0, 2, 100)

# Note that even in the OO-style, we use `.pyplot.figure` to create the figure.
fig, ax = plt.subplots(figsize=(12, 8))  # Create a figure and an axes.
ax.plot(x, x, label='linear')  # Plot some data on the axes.
ax.plot(x, x**2, label='quadratic')  # Plot more data on the axes...
ax.plot(x, x**3, label='cubic')  # ... and some more.
ax.set_xlabel('x label')  # Add an x-label to the axes.
ax.set_ylabel('y label')  # Add a y-lacat_body_tempbel to the axes.
ax.set_title("Simple Plot")  # Add a title to the axes.
ax.legend()  # Add a legend.
```

[12]:
```
<matplotlib.legend.Legend at 0x7f316dcfbf50>
```



7 Histogram

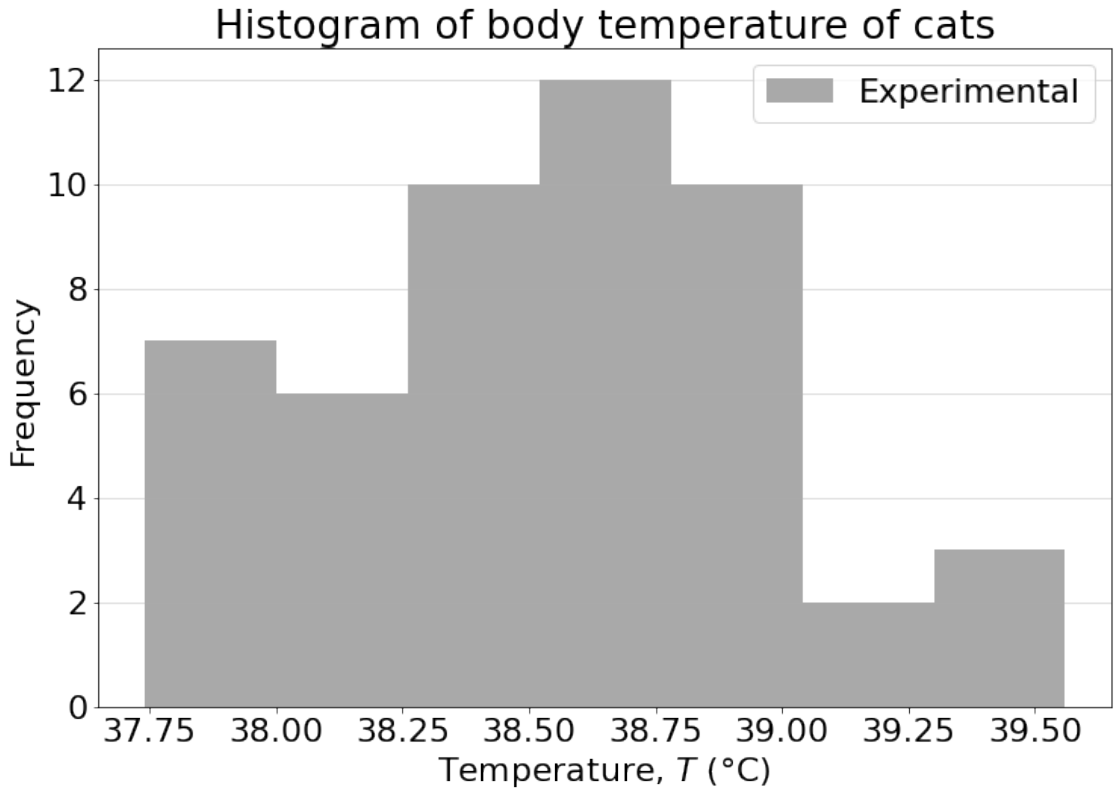
7.1 The dataset

7.1.1 Cat body temperature

[13]:
```
# Data from master course Bayesian Statistics (Emile Apol)
cat_body_temp = [39.54, 37.87, 39.02, 38.30, 38.03, 39.27, 38.68, 38.28, 38.39,␣
↪38.66,
                 39.47, 38.57, 38.93, 38.56, 38.24, 38.91, 38.28, 38.78, 38.50,␣
↪38.07,
                 37.75, 38.68, 37.74, 38.78, 37.83, 39.09, 37.93, 37.92, 39.03,␣
↪38.92,
                 38.72, 39.03, 38.52, 38.97, 38.42, 38.72, 38.77, 38.44, 38.14,␣
↪37.86,
                 38.27, 38.63, 38.35, 38.83, 38.10, 38.54, 38.55, 38.32, 39.56,␣
↪38.22]
```

[14]:
```
# Code adapted from master course Bayesian Statistics (Emile Apol)
plt.figure(figsize=(12, 8))
plt.hist(x=cat_body_temp,
```

```
        bins='auto',
        color='darkgrey',alpha=1, rwidth=1, label='Experimental')
plt.grid(axis='y', alpha=0.5)
plt.xlabel('Temperature, $T$ (\xb0C)')
plt.ylabel('Frequency')
plt.title('Histogram of body temperature of cats')
plt.legend(loc='best')
plt.show()
```

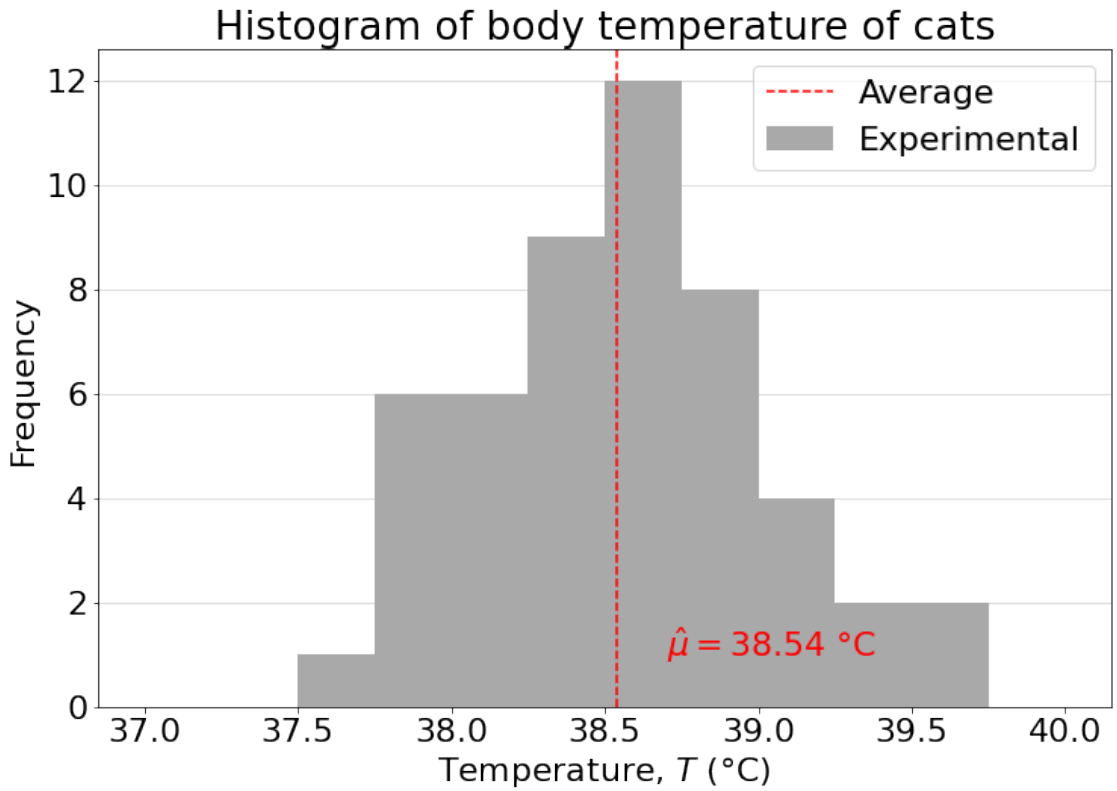


[15]:
```
bins = [37.0, 37.25, 37.5, 37.75, 38.0, 38.25, 38.5, 38.75, 39.0, 39.25, 39.5,␣
↪39.75, 40.0]
```

[16]:
```
import statistics as stat

mean = stat.mean(cat_body_temp)
```

[17]:
```
# Code adapted from master course Bayesian Statistics (Emile Apol)
plt.figure(figsize=(12, 8))
plt.hist(x=cat_body_temp,
        bins=bins,
        color='darkgrey',alpha=1, rwidth=1, label='Experimental')
plt.axvline(x = mean, color = 'red', linestyle='--', label='Average')
```

```
plt.text(38.7, 1.2, '$\^\mu = ${:.2f} \xb0C'.format(mean),␣
↪verticalalignment='center', color='red')
plt.grid(axis='y', alpha=0.5)
plt.xlabel('Temperature, $T$ (\xb0C)')
plt.ylabel('Frequency')
plt.title('Histogram of body temperature of cats')
plt.legend(loc='best')
plt.show()
```



8 Categorical data

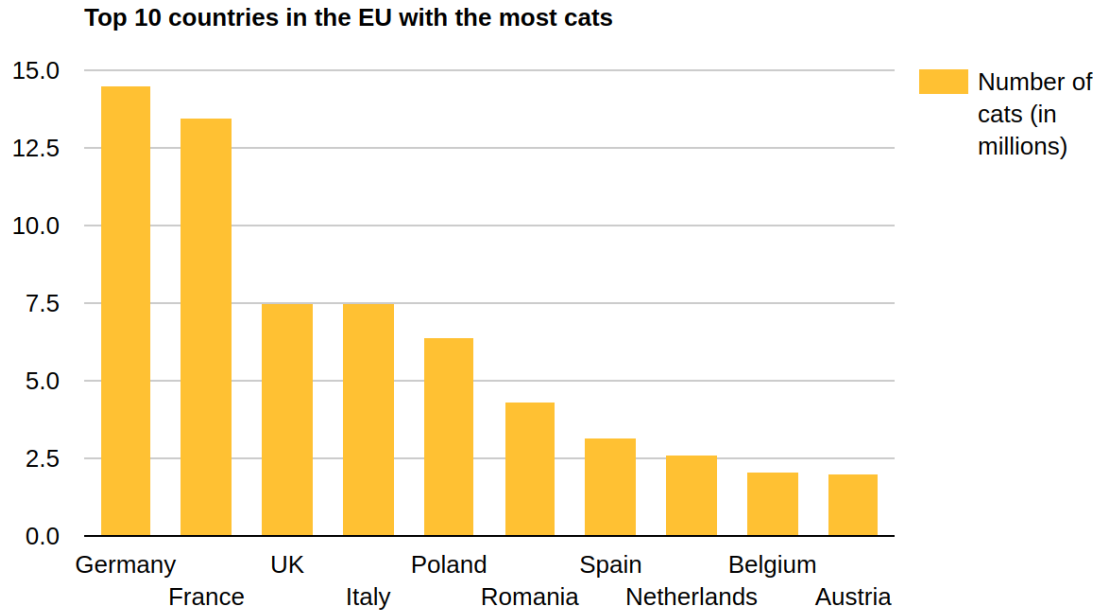
8.1 The dataset

8.1.1 Countries with the most pet cats in the EU

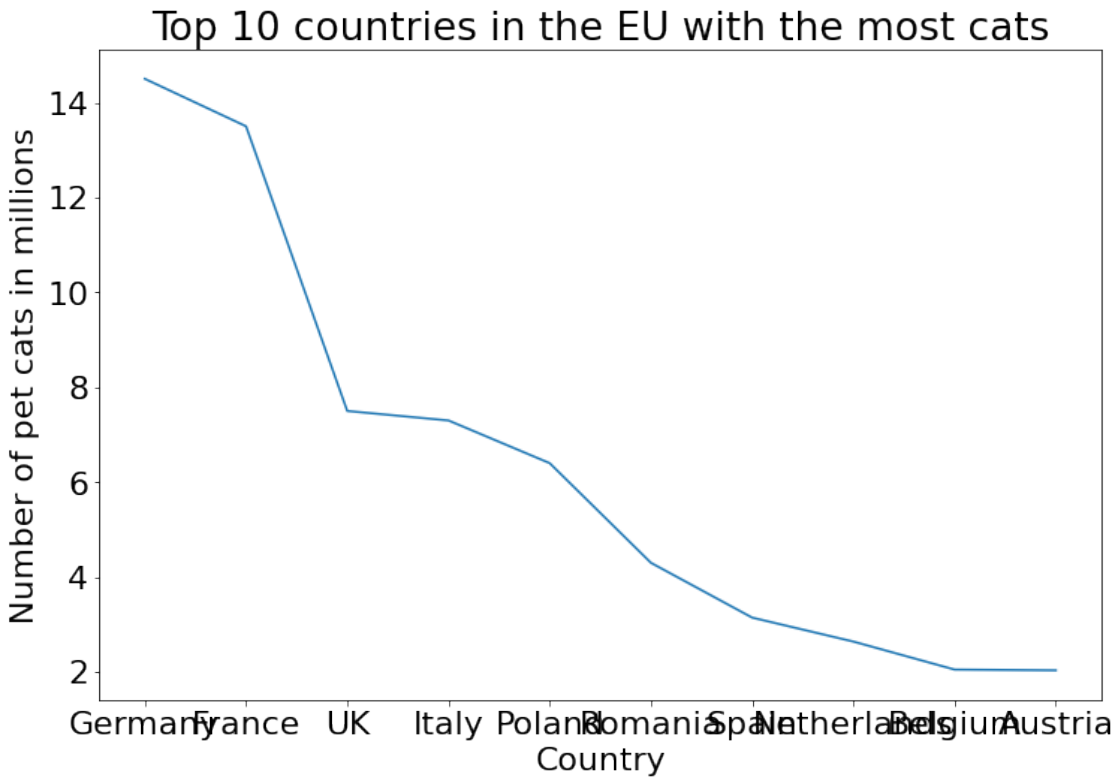
[18]:
```
# data from https://www.worldatlas.com/articles/
↪10-countries-in-the-european-union-with-most-cats.html
countries = ["Germany", "France", "UK", "Italy", "Poland", "Romania", "Spain",␣
↪"Netherlands", "Belgium", "Austria"]
number_of_cats_in_millions = [14.5, 13.5, 7.5, 7.3, 6.4, 4.3, 3.145, 2.640, 2.
↪050, 2.034]
```

[19]:
```
print(len(countries) == len(number_of_cats_in_millions))
```

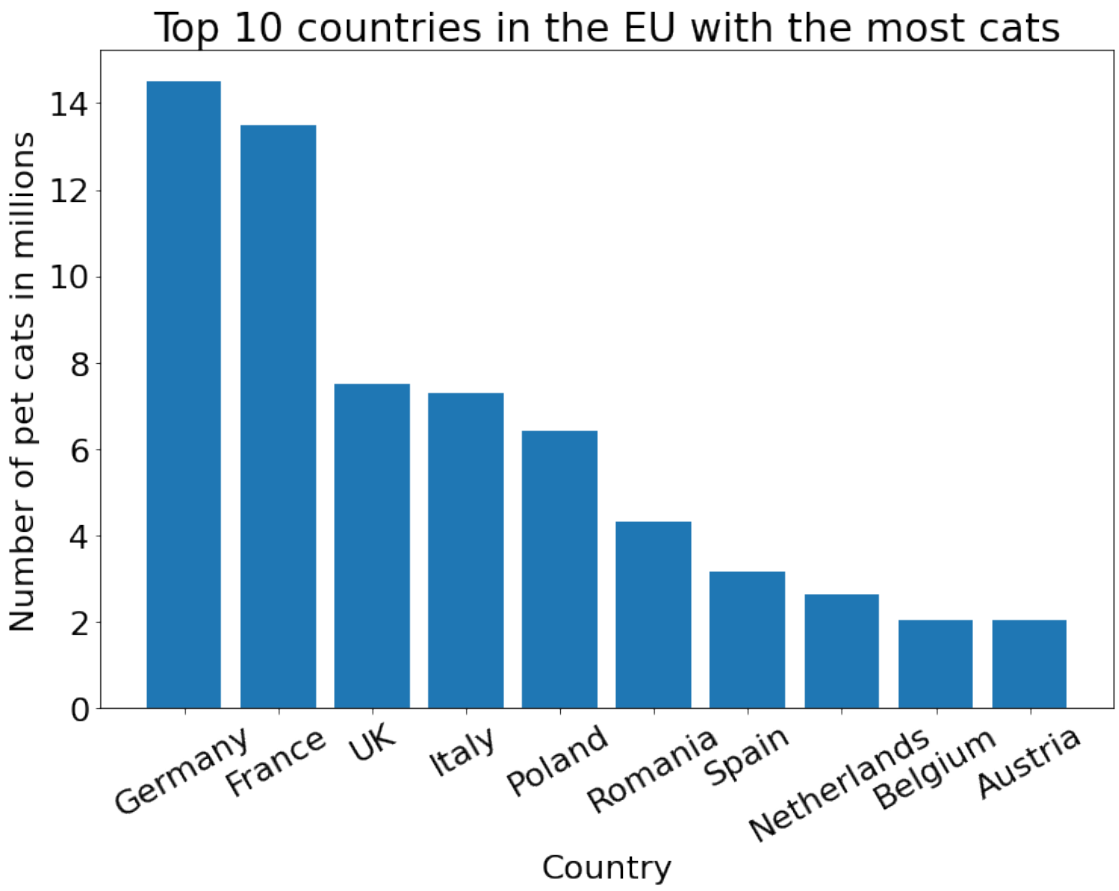
```
True
```



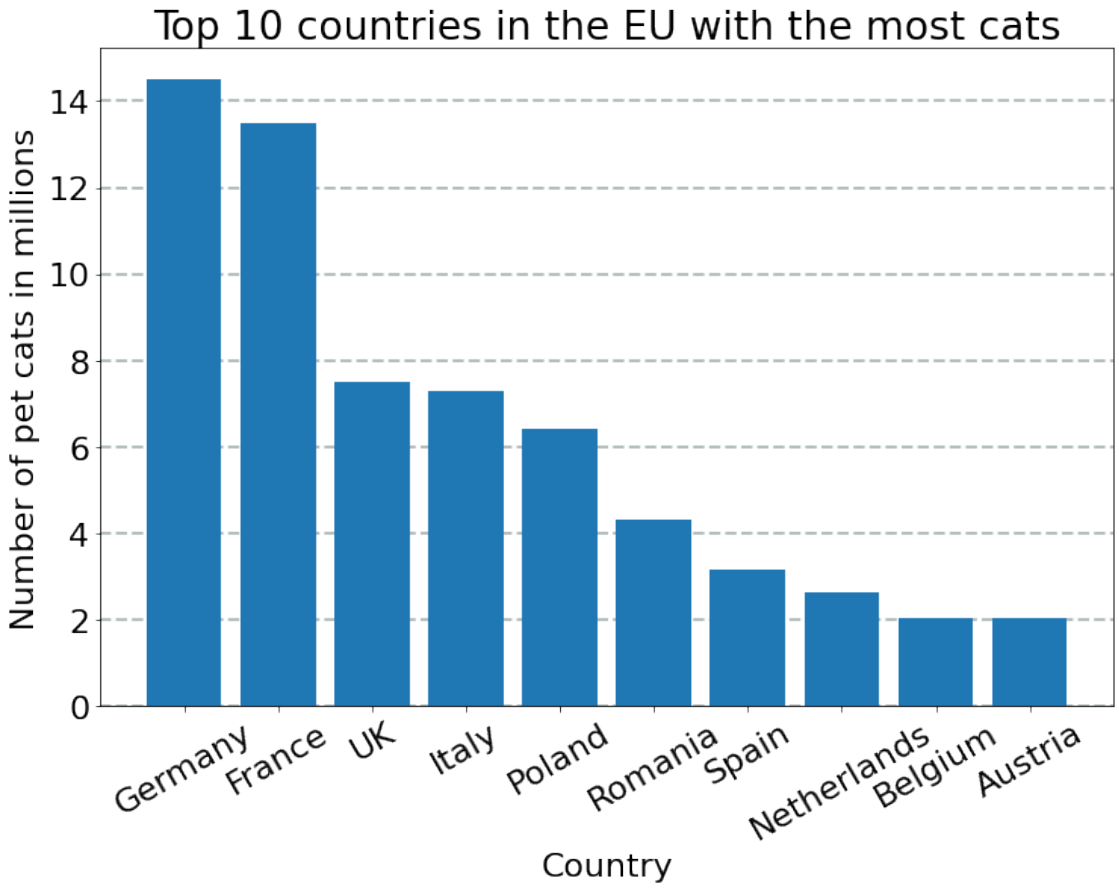
[20]:
```
plt.figure(figsize=(12, 8))
plt.plot(countries, number_of_cats_in_millions)
# plt.bar(countries, number_of_cats_in_millions)
plt.xlabel('Country')
plt.ylabel('Number of pet cats in millions')
plt.title("Top 10 countries in the EU with the most cats")
plt.show()
```



[21]:
```
plt.figure(figsize=(12, 8))
plt.bar(countries, number_of_cats_in_millions)
plt.xticks(rotation=30)
plt.xlabel('Country')
plt.ylabel('Number of pet cats in millions')
plt.title("Top 10 countries in the EU with the most cats")
plt.show()
```



[22]:
```
plt.figure(figsize=(12, 8))
plt.bar(countries, number_of_cats_in_millions, zorder=3)
plt.grid(color='#95a5a6', linestyle='--', linewidth=2, axis='y', alpha=0.7,␣
 ↪zorder=0)
plt.xticks(rotation=30)
plt.xlabel('Country')
plt.ylabel('Number of pet cats in millions')
plt.title("Top 10 countries in the EU with the most cats")
plt.show()
```



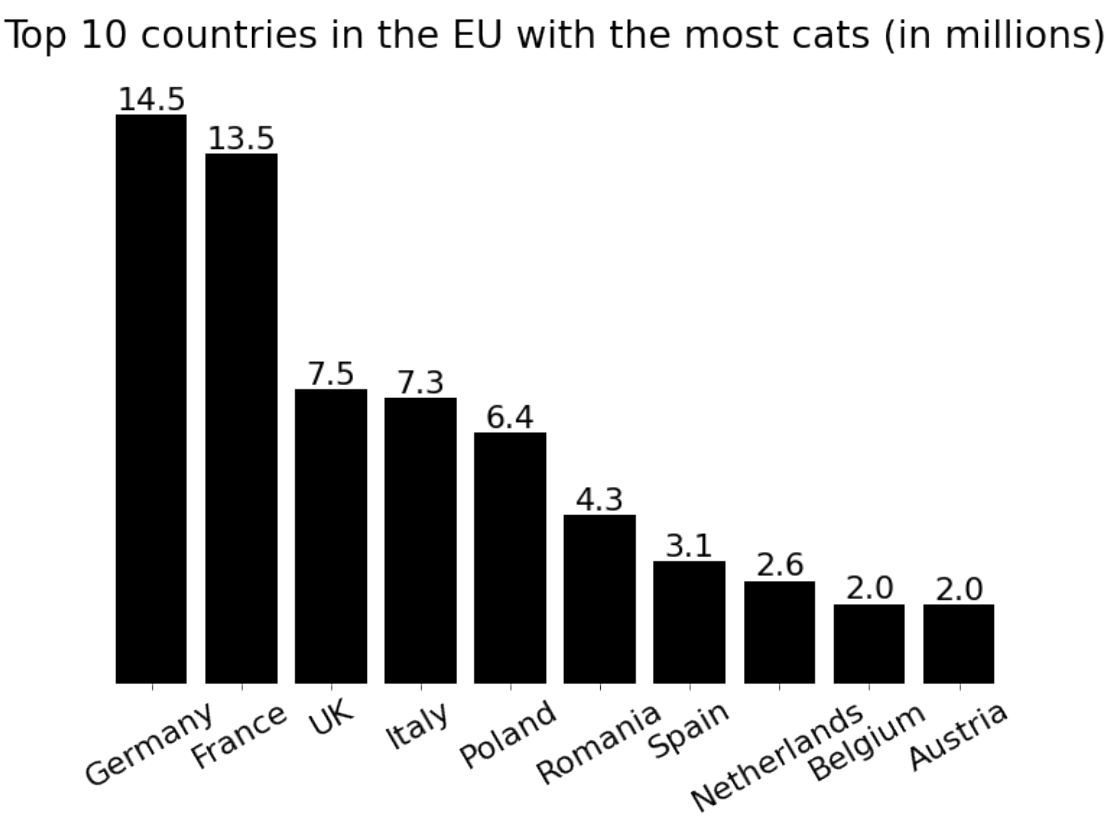
9 Data-Ink ratio

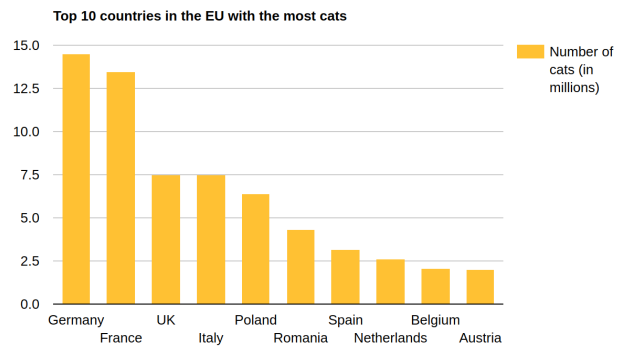
$$\text{Data-Ink ratio} = \frac{\text{Data-Ink}}{\text{Total ink used to print the graphic}}$$

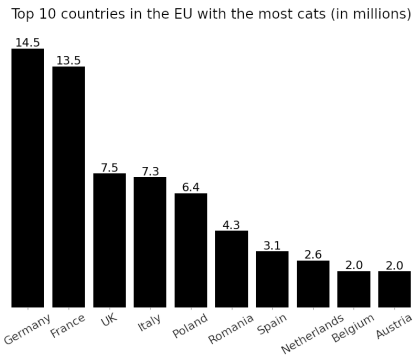
[23]:
```
def plot_pet_cats_globally():
    fig, ax = plt.subplots(figsize=(12, 8))
    rects = ax.bar(countries, number_of_cats_in_millions, color='black')

    for rect in rects:
            height = rect.get_height()
            ax.annotate(xy=(rect.get_x() + rect.get_width()/2., height),
                        text=round(height, 1), ha='center', va='bottom')

    for spine in ax.spines:
        ax.spines[spine].set_visible(False)

    ax.set(title="Top 10 countries in the EU with the most cats (in millions)",
           yticklabels=[], ylim=(0,16))
```

```
    ax.tick_params(left=False)

    plt.xticks(rotation=30)
    return plt
```

[24]:
```
plt.show(plot_pet_cats_globally())
```



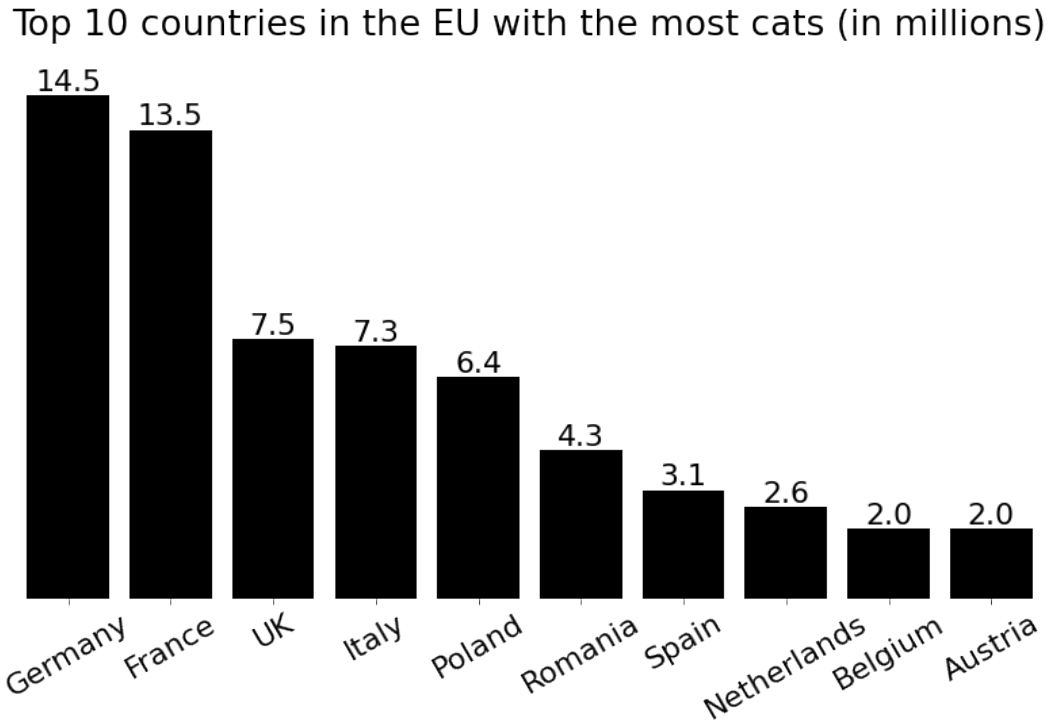




[25]:
```
import base64
import io
```

```
def save_fig_to_html(fig):
    fig.tight_layout()

    pic_IObytes = io.BytesIO()
    fig.savefig(pic_IObytes, format='png')
    pic_IObytes.seek(0)

    pic_hash = base64.b64encode(pic_IObytes.read())

    image_html =f'<img src=\'{pic_hash.
 ↪decode("utf-8")}\'>'

    with open('example.html', 'w') as f:
            f.write(image_html)
```

[26]:
```
fig = plot_pet_cats_globally()
save_fig_to_html(fig)
```



10 For further reading and used sources:

- https://matplotlib.org/stable/tutorials/introductory/usage.html
- https://matplotlib.org/stable/tutorials/introductory/pyplot.html