Exercises chapter 15: Generating data

Saul SL

June 2023

1 Setup

```python
import matplotlib.pyplot as plt
```

2 Exercise 15-1 Cubes

A number raised to the third power is a cube. Plot the first five cubic numbers, and then plot the first 5000 cubic numbers.

```python
i_x = list(range(1, 6))
i_y = [y**3 for y in i_x]

fig, ax = plt.subplots()
ax.plot(i_x, i_y, linewidth=3)

# customization
ax.set_title("Squares", fontsize=18)
ax.set_xlabel("X", fontsize=14)
ax.set_ylabel("x^2", fontsize=14)

plt.show()

i_x = list(range(1, 5001))
i_y = [y**3 for y in i_x]

fig, ax = plt.subplots()
ax.plot(i_x, i_y, linewidth=3)

# customization
ax.set_title("Squares", fontsize=18)
ax.set_xlabel("X", fontsize=14)
ax.set_ylabel("x^2", fontsize=14)

plt.show()
```

3 Exercise 15-2 Colored Cubes

Apply a colormap to your cubes plot.

```python
fig, ax = plt.subplots()
ax.scatter(i_x, i_y, c=i_y, cmap=plt.cm.viridis, s=10)
ax.scatter(i_x, i_y, c=i_y, cmap=plt.cm.PuRd, s=10)
ax.scatter(i_x, i_y, c=i_y, cmap=plt.cm.summer, s=10)
plt.show()
```

4 Exercise 15-3 Molecular Motion

Modify `rw_visual.py` by replacing `plt.scatter()` with `plt.plot()`. To simulate the path of a pollen grain on the surface of a drop of water, pass in the rw.x_values and rw.y_values, and include a linewidth argu- ment. Use 5000 instead of 50,000 points.

```python
from random_walk import RandomWalk

rw = RandomWalk()
rw.fill_walk()

plt.style.use("classic")
fig, ax = plt.subplots(figsize=(15, 9), dpi=128)  # full screen
ax.plot(rw.x_values, rw.y_values, linewidth=3)

# Emphasize the first and last points.
ax.scatter(0, 0, c="black", edgecolors="none", s=100)
ax.scatter(rw.x_values[-1], rw.y_values[-1], c="red", edgecolors="none", s=100)

# Remove the axes.
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)

plt.show()
```

5 Exercise 15-4 Modified Random Walks

In the RandomWalk class, x_step and y_step are generated from the same set of conditions. The direction is chosen randomly from the list [1, -1] and the distance from the list [0, 1, 2, 3, 4]. Modify the values in these lists to see what happens to the overall shape of your walks. Try a longer list of choices for the distance, such as 0 through 8, or remove the −1 from the x or y direction list.

```python
rw = RandomWalk()
rw.fill_walk_v2()

plt.style.use("classic")
fig, ax = plt.subplots(figsize=(15, 9), dpi=128)  # full screen
ax.plot(rw.x_values, rw.y_values, linewidth=2, c="grey")

# Emphasize the first and last points.
ax.scatter(0, 0, c="blue", edgecolors="none", s=100)
ax.scatter(rw.x_values[-1], rw.y_values[-1], c="red", edgecolors="none", s=100)

# Remove the axes.
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)

plt.show()
```

6 Exercise 15-6 Two D8s

Create a simulation showing what happens when you roll two eight-sided dice 1000 times. Try to picture what you think the visualization will look like before you run the simulation; then see if your intuition was correct. Gradually increase the number of rolls until you start to see the limits of your system's capabilities.

```python
idice1 = Dice(8)
idice2 = Dice(8)
n_simul = 10000
```

```
results = []
for _ in range(1, n_simul):
    result = idice1.roll_dice() + idice2.roll_dice()
    results.append(result)

max_result = idice1.sides + idice2.sides
frequencies = []
for result in range(2, max_result + 1):
    frequency = results.count(result)
    frequencies.append(frequency)

x_values = list(range(2, max_result + 1))
data = [Bar(x=x_values, y=frequencies)]
x_axis_config = {"title": "Values", "dtick": 1}
y_axis_config = {"title": "Frequency"}
ilayout = Layout(
    title=f"Results of a {n_simul} simulations",
    xaxis=x_axis_config,
    yaxis=y_axis_config,
)
offline.plot({"data": data, "layout": ilayout}, filename="D8_simulation.html")
```

7 Exercise 15-7 Three Dice

When you roll three D6 dice, the smallest number you can roll is 3 and the largest number is 18. Create a visualization that shows what hap- pens when you roll three D6 dice.

```
idice1 = Dice()
idice2 = Dice()
idice3 = Dice()

n_simul = 10000
results = []
for _ in range(1, n_simul):
    result = idice1.roll_dice() + idice2.roll_dice() + idice3.roll_dice()
    results.append(result)

max_result = idice1.sides + idice2.sides + idice3.sides
frequencies = []
for result in range(3, max_result + 1):
    frequency = results.count(result)
    frequencies.append(frequency)

x_values = list(range(3, max_result + 1))
data = [Bar(x=x_values, y=frequencies)]
x_axis_config = {"title": "Values", "dtick": 1}
y_axis_config = {"title": "Frequency"}
ilayout = Layout(
    title=f"Results of a {n_simul} simulations",
    xaxis=x_axis_config,
    yaxis=y_axis_config,
)
offline.plot({"data": data, "layout": ilayout}, filename="D6_simulation.html")
```

8 Exercise 15-8 Multiplication

When you roll two dice, you usually add the two numbers together to get the result. Create a visualization that shows what happens if you multiply these numbers instead.

```python
idice1 = Dice()
idice2 = Dice()

n_simul = 10000
results = []
for _ in range(1, n_simul):
    result = idice1.roll_dice() * idice2.roll_dice()
    results.append(result)

max_result = idice1.sides * idice2.sides
frequencies = []
for result in range(1, max_result + 1):
    frequency = results.count(result)
    frequencies.append(frequency)

x_values = list(range(1, max_result + 1))
data = [Bar(x=x_values, y=frequencies)]
x_axis_config = {"title": "Values", "dtick": 1}
y_axis_config = {"title": "Frequency"}
ilayout = Layout(
    title=f"Results of a {n_simul} simulations",
    xaxis=x_axis_config,
    yaxis=y_axis_config,
)
offline.plot({"data": data, "layout": ilayout}, filename="D6_product_simulation.html")
```

9 Exercise 15-9 Die Comprehensions

For clarity, the listings in this section use the long form of for loops. If you're comfortable using list comprehensions, try writing a comprehension for one or both of the loops in each of these programs.

```python
idice1 = Dice()
idice2 = Dice()

n_simul = 10000
results = [idice1.roll_dice() * idice2.roll_dice() for i in range(1, n_simul)]

max_result = idice1.sides * idice2.sides

frequencies = [results.count(i) for i in range(1, max_result + 1)]

x_values = list(range(1, max_result + 1))
data = [Bar(x=x_values, y=frequencies)]
x_axis_config = {"title": "Values", "dtick": 1}
y_axis_config = {"title": "Frequency"}
ilayout = Layout(
    title=f"Results of a {n_simul} simulations",
    xaxis=x_axis_config,
    yaxis=y_axis_config,
)
offline.plot(
    {"data": data, "layout": ilayout}, filename="D6_product_simulation_v2.html"
)
```

10 Exercise 15-10 Practicing with Both Libraries

Try using Matplotlib to make a die-rolling visualization, and use Plotly to make the visualization for a random walk. (You'll need to consult the documentation for each library to complete this exercise.)

10.1 matplotlib

```python
import matplotlib.pyplot as plt
from dice import Dice


idice1 = Dice()
idice2 = Dice()

n_simul = 10000
results = [idice1.roll_dice() * idice2.roll_dice() for i in range(1, n_simul)]

max_result = idice1.sides * idice2.sides
frequencies = [results.count(i) for i in range(1, max_result + 1)]
x_values = list(range(1, max_result + 1))

plt.figure(figsize=(15, 9), dpi=128)
plt.bar(x_values, frequencies)
plt.title(f"Results of a {n_simul} simulations", fontsize=20)
plt.xlabel("Values", color="gray")
plt.xticks(list(range(1, max_result + 1)))
plt.ylabel("Frequency", color="gray")
plt.show()
```

10.2 plotly

```python
from plotly.graph_objs import Scatter, Figure, Layout
from plotly import offline
from random_walk import RandomWalk

rw = RandomWalk()
rw.fill_walk()

# Define the marker properties
# imarker = {
#     'size': [10 if i in [0, len(rw.x_values) -1] else 6 for i in range(len(rw.x_values))],  # Customize the size for each poi
#     'color': ['black' if i == 0 else 'red' if i == len(rw.x_values) - 1 else 'blue' for i in range(len(rw.x_values))],  # Cus
#     'symbol': 'circle',  # Use a circle symbol for all points
#     'line': {'width': 0}  # Remove the marker edge
# }

# data = [Scatter(x=rw.x_values,
#                 y=rw.y_values,
#                 mode='markers',
#                 marker=imarker)]

# Define the marker properties
imarker = {
    "size": 15,  # Customize the size for each point
    "color": ["black", "red"],  # customize color for first and last
    "symbol": ["circle", "square"],  # Use a circle symbol for all points
    "line": {"width": 0},  # Remove the marker edge
}

data = [
    Scatter(
        x=rw.x_values,
        y=rw.y_values,
        mode="markers",
        marker={"color": rw.x_values, "colorscale": "Viridis"},
    ),
    Scatter(
        x=[rw.x_values[i] for i in [0, len(rw.x_values) - 1]],
        y=[rw.y_values[i] for i in [0, len(rw.y_values) - 1]],
        mode="markers",
        marker=imarker,
    ),
]

# Remove the axes.
i_layout = Layout(
    xaxis={"visible": False},  # Remove the x-axis
    yaxis={"visible": False},  # Remove the y-axis
)

offline.plot({"data": data, "layout": i_layout}, filename="random_walk_plotly_v3.html")
```