

Exercises chapter 15: Generating data

Saul SL

June 2023

1 Setup

```
1 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.

```
1 i_x = list(range(1, 6))
2 i_y = [y**3 for y in i_x]
3
4 fig, ax = plt.subplots()
5 ax.plot(i_x, i_y, linewidth=3)
6
7 # customization
8 ax.set_title("Squares", fontsize=18)
9 ax.set_xlabel("X", fontsize=14)
10 ax.set_ylabel("x^2", fontsize=14)
11
12 plt.show()
13
14 i_x = list(range(1, 5001))
15 i_y = [y**3 for y in i_x]
16
17 fig, ax = plt.subplots()
18 ax.plot(i_x, i_y, linewidth=3)
19
20 # customization
21 ax.set_title("Squares", fontsize=18)
22 ax.set_xlabel("X", fontsize=14)
23 ax.set_ylabel("x^2", fontsize=14)
24
25 plt.show()
```

3 Exercise 15-2 Colored Cubes

Apply a colormap to your cubes plot.

```
1 fig, ax = plt.subplots()
2 ax.scatter(i_x, i_y, c=i_y, cmap=plt.cm.viridis, s=10)
3 ax.scatter(i_x, i_y, c=i_y, cmap=plt.cm.PuRd, s=10)
4 ax.scatter(i_x, i_y, c=i_y, cmap=plt.cm.summer, s=10)
5 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` argument. Use 5000 instead of 50,000 points.

```
1  from random_walk import RandomWalk
2
3  rw = RandomWalk()
4  rw.fill_walk()
5
6  plt.style.use("classic")
7  fig, ax = plt.subplots(figsize=(15, 9), dpi=128) # full screen
8  ax.plot(rw.x_values, rw.y_values, linewidth=3)
9
10 # Emphasize the first and last points.
11 ax.scatter(0, 0, c="black", edgecolors="none", s=100)
12 ax.scatter(rw.x_values[-1], rw.y_values[-1], c="red", edgecolors="none", s=100)
13
14 # Remove the axes.
15 ax.get_xaxis().set_visible(False)
16 ax.get_yaxis().set_visible(False)
17
18 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.

```
1  rw = RandomWalk()
2  rw.fill_walk_v2()
3
4  plt.style.use("classic")
5  fig, ax = plt.subplots(figsize=(15, 9), dpi=128) # full screen
6  ax.plot(rw.x_values, rw.y_values, linewidth=2, c="grey")
7
8  # Emphasize the first and last points.
9  ax.scatter(0, 0, c="blue", edgecolors="none", s=100)
10 ax.scatter(rw.x_values[-1], rw.y_values[-1], c="red", edgecolors="none", s=100)
11
12 # Remove the axes.
13 ax.get_xaxis().set_visible(False)
14 ax.get_yaxis().set_visible(False)
15
16 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.

```
1  idice1 = Dice(8)
2  idice2 = Dice(8)
3  n_simul = 10000
```

```

4 results = []
5 for _ in range(1, n_simul):
6     result = idice1.roll_dice() + idice2.roll_dice()
7     results.append(result)
8
9 max_result = idice1.sides + idice2.sides
10 frequencies = []
11 for result in range(2, max_result + 1):
12     frequency = results.count(result)
13     frequencies.append(frequency)
14
15 x_values = list(range(2, max_result + 1))
16 data = [Bar(x=x_values, y=frequencies)]
17 x_axis_config = {"title": "Values", "dtick": 1}
18 y_axis_config = {"title": "Frequency"}
19 ilayout = Layout(
20     title=f"Results of a {n_simul} simulations",
21     xaxis=x_axis_config,
22     yaxis=y_axis_config,
23 )
24 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 happens when you roll three D6 dice.

```

1 idice1 = Dice()
2 idice2 = Dice()
3 idice3 = Dice()
4
5 n_simul = 10000
6 results = []
7 for _ in range(1, n_simul):
8     result = idice1.roll_dice() + idice2.roll_dice() + idice3.roll_dice()
9     results.append(result)
10
11 max_result = idice1.sides + idice2.sides + idice3.sides
12 frequencies = []
13 for result in range(3, max_result + 1):
14     frequency = results.count(result)
15     frequencies.append(frequency)
16
17 x_values = list(range(3, max_result + 1))
18 data = [Bar(x=x_values, y=frequencies)]
19 x_axis_config = {"title": "Values", "dtick": 1}
20 y_axis_config = {"title": "Frequency"}
21 ilayout = Layout(
22     title=f"Results of a {n_simul} simulations",
23     xaxis=x_axis_config,
24     yaxis=y_axis_config,
25 )
26 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.

```

1  idice1 = Dice()
2  idice2 = Dice()
3
4  n_simul = 10000
5  results = []
6  for _ in range(1, n_simul):
7      result = idice1.roll_dice() * idice2.roll_dice()
8      results.append(result)
9
10 max_result = idice1.sides * idice2.sides
11 frequencies = []
12 for result in range(1, max_result + 1):
13     frequency = results.count(result)
14     frequencies.append(frequency)
15
16 x_values = list(range(1, max_result + 1))
17 data = [Bar(x=x_values, y=frequencies)]
18 x_axis_config = {"title": "Values", "dtick": 1}
19 y_axis_config = {"title": "Frequency"}
20 ilayout = Layout(
21     title=f"Results of a {n_simul} simulations",
22     xaxis=x_axis_config,
23     yaxis=y_axis_config,
24 )
25 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.

```

1  idice1 = Dice()
2  idice2 = Dice()
3
4  n_simul = 10000
5  results = [idice1.roll_dice() * idice2.roll_dice() for i in range(1, n_simul)]
6
7  max_result = idice1.sides * idice2.sides
8
9  frequencies = [results.count(i) for i in range(1, max_result + 1)]
10
11 x_values = list(range(1, max_result + 1))
12 data = [Bar(x=x_values, y=frequencies)]
13 x_axis_config = {"title": "Values", "dtick": 1}
14 y_axis_config = {"title": "Frequency"}
15 ilayout = Layout(
16     title=f"Results of a {n_simul} simulations",
17     xaxis=x_axis_config,
18     yaxis=y_axis_config,
19 )
20 offline.plot(
21     {"data": data, "layout": ilayout}, filename="D6_product_simulation_v2.html"
22 )

```

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

```
1 import matplotlib.pyplot as plt
2 from dice import Dice
3
4
5 idice1 = Dice()
6 idice2 = Dice()
7
8 n_simul = 10000
9 results = [idice1.roll_dice() * idice2.roll_dice() for i in range(1, n_simul)]
10
11 max_result = idice1.sides * idice2.sides
12 frequencies = [results.count(i) for i in range(1, max_result + 1)]
13 x_values = list(range(1, max_result + 1))
14
15 plt.figure(figsize=(15, 9), dpi=128)
16 plt.bar(x_values, frequencies)
17 plt.title(f"Results of a {n_simul} simulations", fontsize=20)
18 plt.xlabel("Values", color="gray")
19 plt.xticks(list(range(1, max_result + 1)))
20 plt.ylabel("Frequency", color="gray")
21 plt.show()
```

10.2 plotly

```
1 from plotly.graph_objs import Scatter, Figure, Layout
2 from plotly import offline
3 from random_walk import RandomWalk
4
5 rw = RandomWalk()
6 rw.fill_walk()
7
8 # Define the marker properties
9 imarker = {
10     # '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 point
11     # 'color': ['black' if i == 0 else 'red' if i == len(rw.x_values) - 1 else 'blue' for i in range(len(rw.x_values))], # Customize the color for each point
12     # 'symbol': 'circle', # Use a circle symbol for all points
13     # 'line': {'width': 0} # Remove the marker edge
14 }
15
16 # data = [Scatter(x=rw.x_values,
17 #                 y=rw.y_values,
18 #                 mode='markers',
19 #                 marker=imarker)]
20
21 # Define the marker properties
22 imarker = {
23     "size": 15, # Customize the size for each point
24     "color": ["black", "red"], # customize color for first and last
25     "symbol": ["circle", "square"], # Use a circle symbol for all points
26     "line": {"width": 0}, # Remove the marker edge
27 }
28
29 data = [
30     Scatter(
31         x=rw.x_values,
32         y=rw.y_values,
33         mode="markers",
34         marker={"color": rw.x_values, "colorscale": "Viridis"},
```

```

35     ),
36     Scatter(
37         x=[rw.x_values[i] for i in [0, len(rw.x_values) - 1]],
38         y=[rw.y_values[i] for i in [0, len(rw.y_values) - 1]],
39         mode="markers",
40         marker=imarker,
41     ),
42 ]
43
44 # Remove the axes.
45 i_layout = Layout(
46     xaxis={"visible": False}, # Remove the x-axis
47     yaxis={"visible": False}, # Remove the y-axis
48 )
49
50 offline.plot({"data": data, "layout": i_layout}, filename="random_walk_plotly_v3.html")

```