TITLE: Analyzing Distributions: Rolling Dice with Plotly

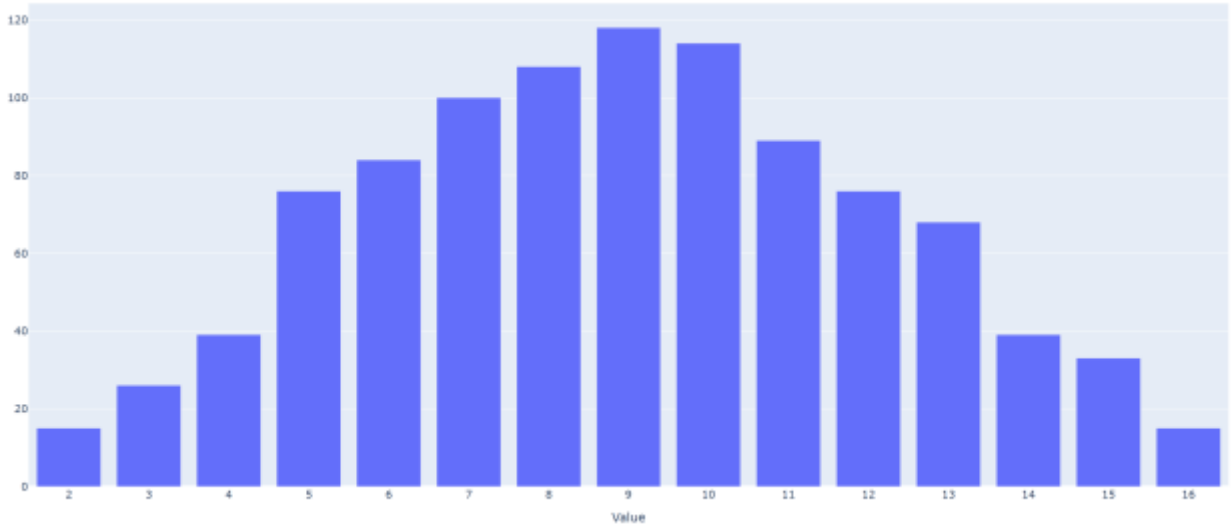
SUBTITLE: Guidebook 3: Interactive Histograms, Probability, and The Bell Curve

SERIES: *Python Data Visualization Series (Book 3 of 3)*

AUTHOR: Faizan Toheed

DATE: January 19, 2026

ABSTRACT: While Matplotlib is excellent for lines and scatter plots, this guide introduces **Plotly**—a library designed for data analytics and interactive dashboards. You will learn to build a simulation engine for rolling dice, analyze the resulting frequencies, and generate professional Bar Charts (Histograms) as HTML files. The guide concludes by visualizing the famous "Bell Curve" distribution by simulating multiple dice.



The Concept: Why Plotly?

In the previous guides, we created static images (PNGs). However, when analyzing statistical data—like how often a number appears—we often want to see the exact count. **Plotly** generates interactive HTML files. This allows you to:

- Hover over a bar to see the exact frequency.
- Zoom in on specific sections of the graph.
- Resize the graph dynamically in a web browser.

Step 1: Building the Simulation (Die Class)

We need a digital object that behaves like a real die.

- **Attribute:** num_sides (Default is 6).
- **Method:** roll() which returns a random number between 1 and the number of sides.

```python
from random import randint

class Die:
    """A class representing a single die."""

    def __init__(self, num_sides=6):
        self.num_sides = num_sides

    def roll(self):
        """Return a random value between 1 and num_sides."""
        return randint(1, self.num_sides)
```

Step 2: Generating Data (List Comprehensions)

We want to roll the die 1,000 times. Instead of a 4-line for loop, we use a **List Comprehension** to do it in one line.

```python
# Create a D6
die = Die()

# Roll it 1000 times
results = [die.roll() for _ in range(1000)]
```

Step 3: Analyzing Frequencies (The Math)

Plotly cannot graph the raw list of rolls (e.g., [1, 4, 6, 1, 2...]). It needs to know **how many times** each number appeared. We iterate through the possible numbers (1 to 6) and count them.

```
frequencies = []
for value in range(1, die.num_sides + 1):
    frequency = results.count(value)
    frequencies.append(frequency)

# Result example: [150, 162, 148, 170, 155, 215]
```

Step 4: Visualizing with Plotly (Bar and Layout)

In Plotly, a chart is built using two dictionaries: data (what to draw) and layout (how it looks).

- **The Data:** We use Bar because histograms are essentially bar charts.

```
from plotly.graph_objs import Bar, Layout
from plotly import offline

# x = the numbers (1-6), y = the counts
x_values = list(range(1, die.num_sides + 1))
data = [Bar(x=x_values, y=frequencies)]
```

- **The Layout:** This controls the titles and labels.

```
my_layout = Layout(title='Results of Rolling One D6 1000 Times',
                   xaxis={'title': 'Result'},
                   yaxis={'title': 'Frequency of Result'})
```

Step 5: Enhancement 1 - Formatting the Axes (dtick)

By default, Plotly might label the X-axis as "2, 4, 6", skipping the odd numbers. To force it to show every integer, we use the dtick (Distance of Tick) setting.

```
xaxis={'title': 'Result', 'dtick': 1} # Forces a label for every number
```

Step 6: Enhancement 2 - Rolling Two Dice (The Bell Curve)

If we roll **two** dice and add the numbers, the result changes drastically.

- **Minimum:** 2 ($1+1$)
- **Maximum:** 12 ($6+6$)
- **Most Common:** 7 (Because there are many ways to make 7: $1+6, 2+5, 3+4...$)

This creates a **Bell Curve** shape.

```
# 1. Create two dice
die_1 = Die()
die_2 = Die()

# 2. Add the rolls
results = [die_1.roll() + die_2.roll() for _ in range(1000)]

# 3. Analyze (Max result is 12)
max_result = die_1.num_sides + die_2.num_sides
frequencies = [results.count(value) for value in range(2, max_result+1)]
```

Summary: The Complete Code Block

Here is the final script dice_visual.py that simulates rolling two dice and generates the interactive d6_d6.html file.

```python
from plotly.graph_objs import Bar, Layout
from plotly import offline
from die import Die

# 1. Create two D6 dice
die_1 = Die()
die_2 = Die()

# 2. Make some rolls
results = [die_1.roll() + die_2.roll() for _ in range(1000)]

# 3. Analyze the results
max_result = die_1.num_sides + die_2.num_sides
frequencies = [results.count(value) for value in range(2, max_result+1)]

# 4. Visualize the results
x_values = list(range(2, max_result+1))
data = [Bar(x=x_values, y=frequencies)]

x_axis_config = {'title': 'Result', 'dtick': 1}
y_axis_config = {'title': 'Frequency of Result'}

my_layout = Layout(title='Results of rolling two D6 dice (1000 rolls)',
                   xaxis=x_axis_config, yaxis=y_axis_config)

offline.plot({'data': data, 'layout': my_layout}, filename='d6_d6.html')
```