How to Easily Create a Pareto Chart in Python

Authored by

stats writer

December 3, 2025

ARABPSYCHOLOGY.COM

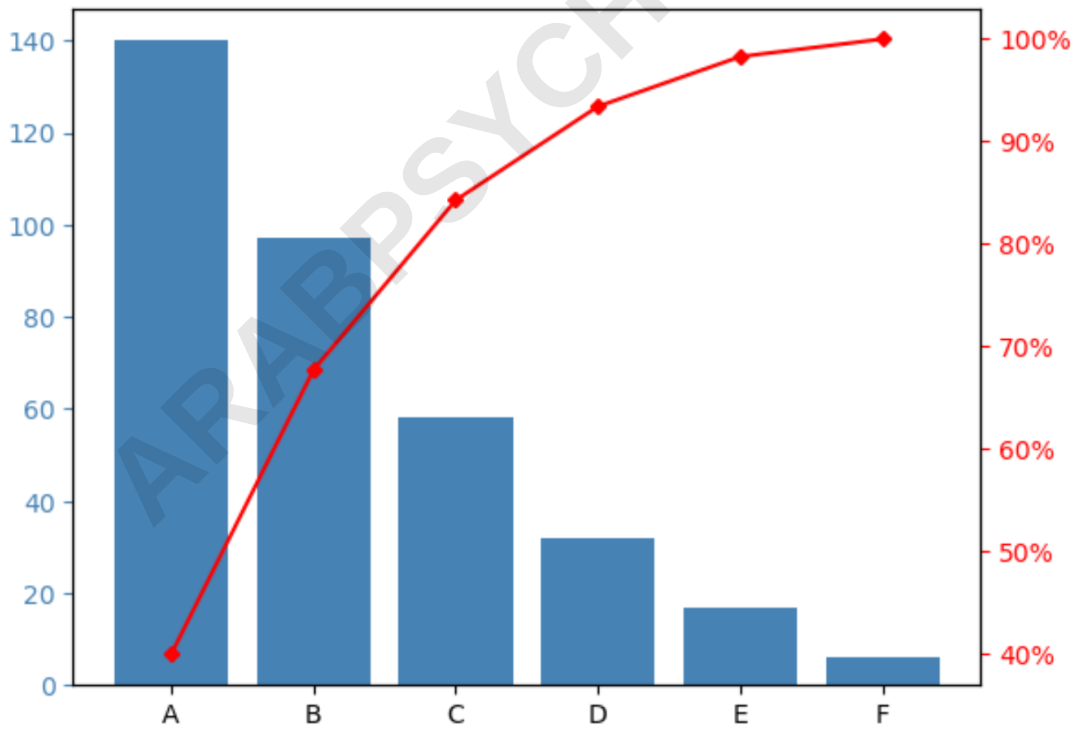
RECOMMENDED CITATION

stats writer (2025). *How to Easily Create a Pareto Chart in Python*. PSYCHOLOGICAL SCALES. Retrieved from https://scales.arabpsychology.com/?p=103949

A Pareto chart is a specialized type of bar graph that is invaluable for quality control and statistical analysis. This powerful visualization tool arranges categorical data in descending order of frequency, simultaneously plotting the cumulative frequency on a secondary axis. This duality makes the Pareto chart an exceptional instrument for identifying the most significant factors within a dataset, often linking directly to the influential 80/20 principle (the Pareto principle).

Creating these detailed charts efficiently requires robust data manipulation and visualization tools. In this comprehensive tutorial, we will utilize the strength of the Python ecosystem, specifically leveraging the **Pandas** library for data preparation and the versatile Matplotlib library for generating the final, high-quality graphical output. We will walk through the process of structuring raw data, calculating the necessary cumulative percentages, and applying appropriate visual aesthetics to produce a clear, informative Pareto chart.

A **Pareto chart** is fundamentally a combination graph, integrating both bars and a line plot. The bars represent the raw frequencies of categories, sorted from the largest occurrence down to the smallest. The superimposed line plot displays the running total, or the cumulative percentage, allowing analysts to quickly determine which small group of causes contributes to the majority of the total effect--a central tenet of quality management and statistical process control.



ARABPSYCHOLOGY.COM

This tutorial provides a detailed, step-by-step methodology, complete with executable code snippets, demonstrating exactly how to generate a precise and customized Pareto chart within the

Python programming environment, ensuring adherence to statistical best practices.

Prerequisites: Essential Python Libraries

Before diving into the data analysis and visualization process, it is essential to ensure that the necessary libraries are installed in your Python environment. For this specific task, we rely on two foundational libraries: **Pandas**, which excels at data structuring and cleaning, and **Matplotlib**, which provides the plotting framework. If you are working in a new environment, these libraries can typically be installed using a package manager like pip.

The functionality of **Pandas** is crucial for handling the initial survey responses and transforming them into a structured format suitable for sorting and calculation. We need to convert the raw counts into a tabular structure, sort these counts, and then derive the cumulative statistics necessary for the Pareto analysis. Furthermore, **Matplotlib** provides the core functions for creating figures, axes, and rendering both the bar components and the line components of our combined chart.

We will specifically utilize the `pandas.DataFrame` object to hold our data and the `matplotlib.pyplot` module, often imported as `plt`, to manage the plotting process, including the application of dual y-axes necessary for displaying both raw counts and cumulative percentages simultaneously.

Step 1: Preparing and Structuring the Data in Pandas

To illustrate the creation of the chart, let us establish a realistic dataset. We will simulate the results of a consumer survey where 350 individuals were asked to identify their preferred cereal brand from a set of six brands (labeled A through F). Properly structuring this raw count data into a Pandas DataFrame is the critical first stage of the analysis, as the Pareto methodology requires precise sorting and aggregation before visualization can begin.

The initial setup involves creating the DataFrame with the counts and assigning the corresponding brand labels as the index. The fundamental requirement for a Pareto chart is that the data must be sorted in descending order of frequency. Therefore, immediately after creation, we use the `sort_values` function within Pandas to arrange the dataset based on the 'count' column, ensuring that Brand A, with the highest frequency, appears first, followed sequentially by the remaining brands.

Following the sorting step, we must introduce the crucial element of the Pareto analysis: the cumulative percentage. This is calculated by first finding the cumulative sum of the counts using the `cumsum()` method and then dividing this running total by the grand total of all counts, multiplying by 100 to express the result as a percentage. This calculation yields the necessary data

ARABPSYCHOLOGY.COM

points for the secondary line plot, providing immediate insight into the contribution of the top categories. Below is the Python code used to prepare and calculate the data:

import pandas as pd

#create DataFrame with raw survey counts
df = pd.DataFrame({'count': })
df.index =

#sort DataFrame by count in descending order
df = df.sort_values(by='count', ascending=False)

#add column to display cumulative percentage
df = df.cumsum()/df.sum()*100

#view DataFrame structure and results
df

count cumperc
A 140 40.000000
B 97 67.714286
C 58 84.285714
D 32 93.428571
E 17 98.285714
F 6 100.000000

Understanding Cumulative Percentage Calculation

The concept of cumulative frequency is central to the efficacy of the Pareto chart. It represents the running total of frequencies as we move down the ranked categories. In our example, the first brand (A) accounts for 140 responses out of 350, resulting in 40% of the total. When we add the responses for the second brand (B), the cumulative count becomes 140 + 97 = 237, which translates to a cumulative percentage of 67.71%.

This progressive aggregation is visually critical because it directly highlights the point at which a specific subset of categories reaches a high threshold, typically the 80% mark, which is often used as a decision point in quality improvement projects. By calculating the `cumperc` column in the Pandas DataFrame, we are creating the dataset that the secondary Matplotlib axis will plot.

It is important to note that the cumulative percentage for the final category (Brand F in this case) must always reach 100%. If the final value deviates, it indicates an error in either the summation or

ARABPSYCHOLOGY.COM

the division by the grand total, necessitating a review of the data preparation steps. This careful preparation ensures the statistical validity of the resulting visualization.

Step 2: Generating the Basic Pareto Chart using Matplotlib

Once the data is correctly structured and the cumulative percentages are calculated in the DataFrame, the next stage involves generating the visualization using the Matplotlib library. The Pareto chart requires two distinct plotting mechanisms sharing the same x-axis (the categories/brands): a bar plot for the counts and a line plot for the cumulative percentages. Matplotlib handles this elegantly using the concept of a **twin axis**.

We begin by defining the plot aesthetics, such as colors for the bars and the line. We then initialize the figure and the primary axis (`ax`) for the bar plot, mapping the brand index (X-axis) against the raw 'count' (left Y-axis). Crucially, we then create the secondary axis (`ax2`) using `ax.twinx()`, which shares the x-axis but provides an independent y-axis on the right side of the plot for the percentage data.

The line plot is drawn on `ax2` using the 'cumperc' data. To ensure that the right-hand axis correctly displays percentage values, we must apply the `PercentFormatter()` from Matplotlib's `ticker` module. Finally, to distinguish the two scales and enhance clarity, we specify the colors for the tick labels on both the primary and secondary y-axes, corresponding to the colors used for the respective plots (blue for bars/counts and red for the line/percentages).

```
import matplotlib.pyplot as plt
from matplotlib.ticker import PercentFormatter

#define aesthetics for plot
color1 = 'steelblue'
color2 = 'red'
line_size = 4

#create basic bar plot on primary axis (ax)
fig, ax = plt.subplots()
ax.bar(df.index, df, color=color1)

#add cumulative percentage line to secondary axis (ax2)
ax2 = ax.twinx()
ax2.plot(df.index, df, color=color2, marker="D", ms=line_size)
ax2.yaxis.set_major_formatter(PercentFormatter())

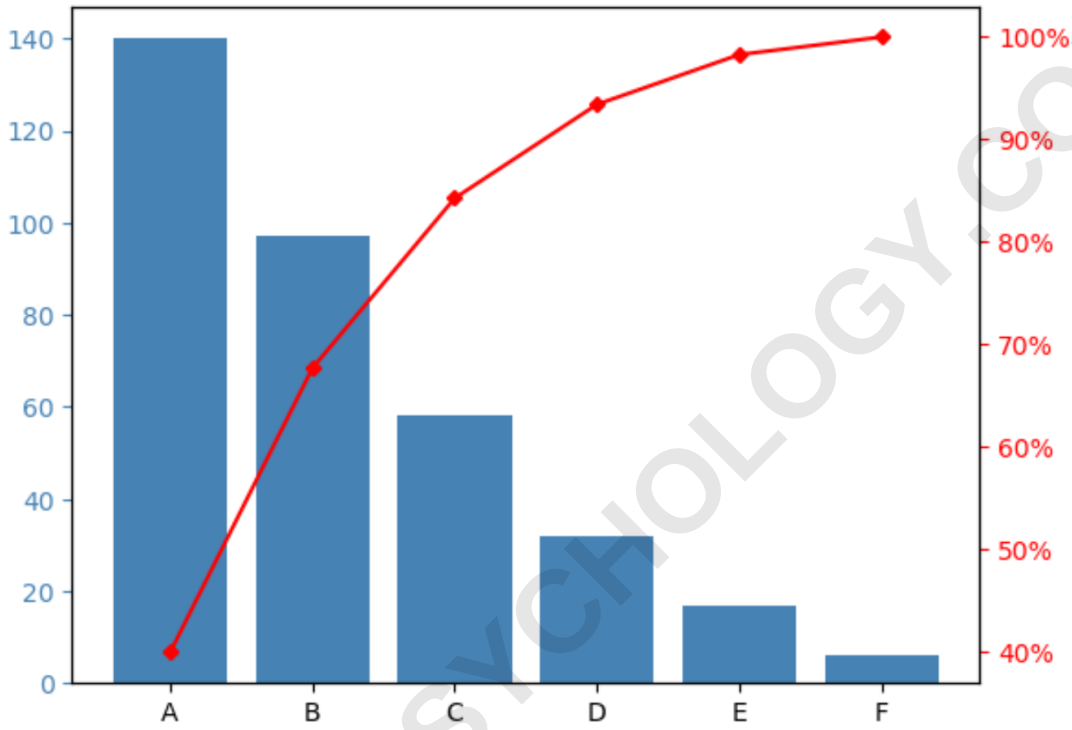
#specify axis colors for clarity
ax.tick_params(axis='y', colors=color1)
```

ARABPSYCHOLOGY.COM

```
ax2.tick_params(axis='y', colors=color2)

#display Pareto chart
plt.show()
```

Interpreting the Visual Results (Analysis)



ARABPSYCHOLOGY.COM

The resulting visual output provides a clear, quantitative assessment of the survey data, adhering to the core principles of the Pareto chart. The horizontal x-axis displays the cereal brands, critically ordered from the highest frequency (Brand A) down to the lowest (Brand F). This sequential ordering is what enables the rapid identification of dominant factors.

The left-hand y-axis corresponds to the blue bars and represents the absolute **frequency** or raw count of responses for each brand. Conversely, the right-hand y-axis corresponds to the red line and indicates the **cumulative percentage**. This line is the key analytical feature, as its curvature reveals the concentration of preference among the top brands.

Analyzing the interaction between the bars and the cumulative line yields profound insights:

Brand A alone accounts for approximately 40% of all total survey responses, confirming it as the most popular choice.
The combination of Brands A and B is highly significant, collectively accounting for nearly 70%

(specifically 67.71%) of total customer preferences.
The top three brands (A, B, and C) solidify the majority, representing roughly 85% (84.28%) of the total market share captured by this survey.

This observation directly supports the Pareto principle, showing that a small number of brands (3 out of 6, or 50% of the categories) account for a vast majority (85%) of the results. This insight is essential for resource allocation, such as prioritizing inventory or marketing efforts toward the dominant brands.

Step 3: Customizing Aesthetics for Enhanced Clarity (Optional)

While the default chart generated in Step 2 is functional, Matplotlib offers extensive options for customization, allowing users to tailor the visual presentation to specific reporting standards or personal preferences. Customization ensures the chart is not only accurate but also visually appealing and maximally clear to the intended audience. For instance, color choices can be adjusted, line thickness can be increased, and marker styles can be changed to improve data differentiation.

To demonstrate this flexibility, we will modify the aesthetic parameters defined in the initial plotting code. We can change the bar color from 'steelblue' to 'pink', the line color from 'red' to 'purple', and increase the line thickness by setting the `line_size` variable from 4 to 6. These minor adjustments can dramatically impact the visual flow and emphasize certain aspects of the data presentation without altering the underlying statistical analysis.

All other core logic--such as the use of `ax.twinx()` and the application of `PercentFormatter()`--remains unchanged, as these are structural requirements of the Pareto chart visualization methodology. This ability to easily adjust aesthetics makes Python a powerful tool for generating production-ready statistical graphics.

```
import matplotlib.pyplot as plt
from matplotlib.ticker import PercentFormatter

#define aesthetics for plot
color1 = 'pink'
color2 = 'purple'
line_size = 6

#create basic bar plot
fig, ax = plt.subplots()
ax.bar(df.index, df, color=color1)

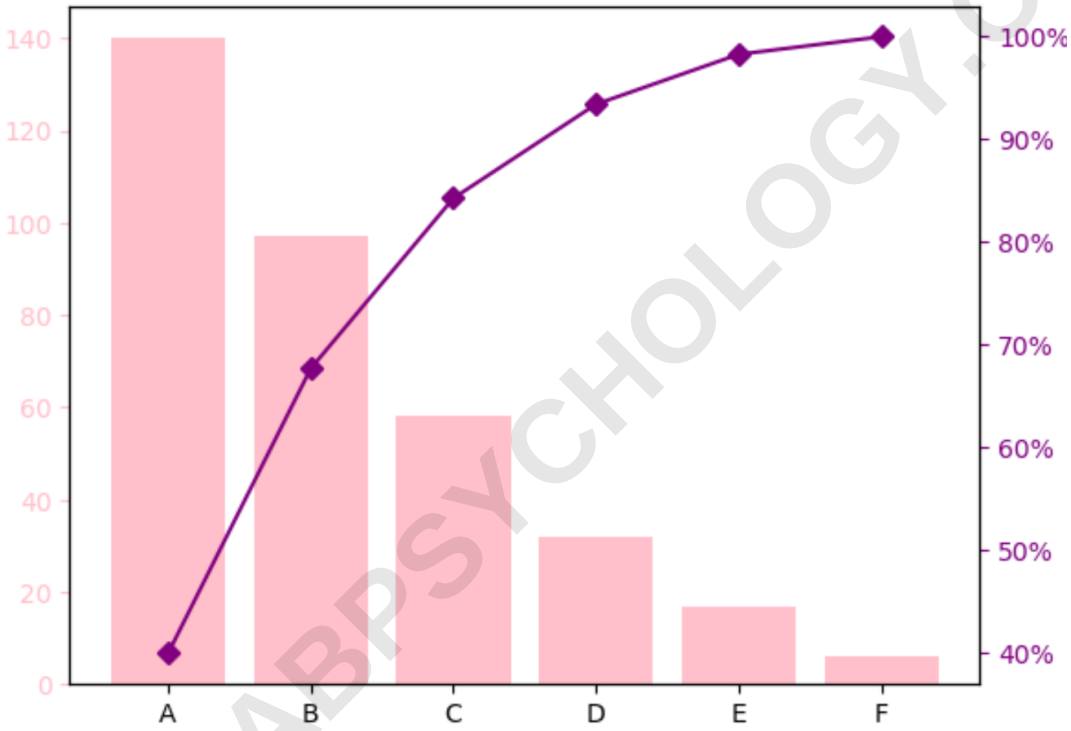
#add cumulative percentage line to plot
```

ARABPSYCHOLOGY.COM

```
ax2 = ax.twinx()
ax2.plot(df.index, df, color=color2, marker="D", ms=line_size)
ax2.yaxis.set_major_formatter(PercentFormatter())

#specify axis colors
ax.tick_params(axis='y', colors=color1)
ax2.tick_params(axis='y', colors=color2)

#display Pareto chart
plt.show()
```



ARABPSYCHOLOGY.COM

Conclusion and Further Applications

The ability to generate a statistically valid and aesthetically controlled Pareto chart using Python, Pandas, and Matplotlib is a fundamental skill for data analysts and quality control professionals. By systematically sorting data, calculating cumulative frequency, and utilizing Matplotlib's dual-axis plotting capabilities, we can quickly translate raw frequency data into actionable insights based on the 80/20 rule.

The methodologies outlined here are broadly applicable beyond simple consumer surveys. Pareto charts are routinely used in manufacturing to identify the most frequent causes of defects, in business to determine the top contributing factors to customer complaints, and in finance to

pinpoint the highest-impact costs. Mastering the creation of this chart in Python allows for scalable, repeatable analysis across various domains, enhancing the decision-making process.

For those interested in exploring further data visualization techniques, Python offers extensive tools for generating other common statistical graphics. Continuing to practice with Matplotlib and the Pandas DataFrame will unlock proficiency in generating everything from histograms and scatter plots to complex box plots and heatmaps, further empowering your data storytelling capabilities.

The following tutorials explain how to create other common visualizations in Python:

ARABPSYCHOLOGY.COM