Learning Pandas: Visualizing Data Distribution with Value Counts

Authored by

Mohammed looti

April 20, 2026

Mastering the distribution of categorical variables is an essential prerequisite for insightful data analysis. The powerful Pandas library, a cornerstone of the scientific computing ecosystem in Python, provides straightforward methods for frequency tabulation and visualization. Central to this process is the **value_counts()** function. This method operates on a Series object (typically a column from a DataFrame) and efficiently returns a Series containing counts of unique values, organized by default in descending order of frequency.

While raw numerical counts offer precision, they often fail to communicate the data narrative effectively. Transforming these frequency distributions into graphical representations significantly enhances comprehension, making patterns, outliers, and skewness immediately apparent. Plotting value counts is therefore not just a supplementary step but a critical component of exploratory data analysis (EDA), streamlining the transition from raw data to actionable insights.

Three Essential Approaches for Plotting Value Counts

Once the frequency distribution has been computed using **value_counts()**, the resulting Pandas Series object inherits convenient plotting methods, streamlining the transition from data tabulation to data visualization. The selection of the plotting method is paramount, as it dictates how the categories are presented--whether sorted by frequency, alphabetically, or based on their original appearance sequence within the dataset.

These visualization strategies primarily utilize the versatile .plot() method available in Pandas Series. Understanding the nuances of each approach ensures that the visual output supports the specific analytical goal, clearly communicating relative dominance or contextual order.

We outline three fundamental techniques for visualizing the output generated by the **value_counts()** function, each serving a distinct purpose in highlighting different aspects of the data distribution:

Plotting in Descending Order (Default): This standard method is employed most often, as it immediately highlights the dominant categories by presenting them from the highest frequency to the lowest. This requires no additional sorting arguments.

df.my_column.value_counts().plot(kind='bar')

Plotting in Ascending Order: To draw attention to categories with lower frequencies--often crucial for identifying rare events or outliers--the results must be explicitly sorted in ascending order. This is accomplished by chaining the **sort_values()** method directly before the plot command.

df.my_column.value_counts().sort_values().plot(kind='bar')

Plotting in Order of Appearance in DataFrame: When the sequence in which categories first appear in the source DataFrame is meaningful (e.g., temporal or sequential data), preserving this order is vital. This requires indexing the results of **value_counts()** using the categories retrieved via the **unique()** function.

df.my_column.value_counts().plot(kind='bar')

Preparing the Sample Dataset for Demonstration

To provide tangible examples of these plotting techniques, we will first establish a small, representative Pandas DataFrame. This dataset, which tracks team membership and associated scores, is intentionally simple, allowing us to clearly demonstrate how different sorting mechanisms affect the resulting visual output when analyzing categorical data.

The primary objective of this setup phase is two-fold: initializing the data structure and performing the initial frequency calculation using the **value_counts()** method on the target column, 'team'. This initial calculation confirms the raw numerical distribution, which the subsequent plots will visualize.

import pandas as pd

# Create the sample DataFrame
df = pd.DataFrame({'team': ,
'points': })

# Display the DataFrame structure
print(df)

team points
0 A 15
1 A 12
2 B 18
3 B 20
4 B 22
5 B 28
6 B 35
7 C 40

# Calculate occurrences of each value in the 'team' column
df.team.value_counts()

B 5
A 2
C 1
Name: team, dtype: int64

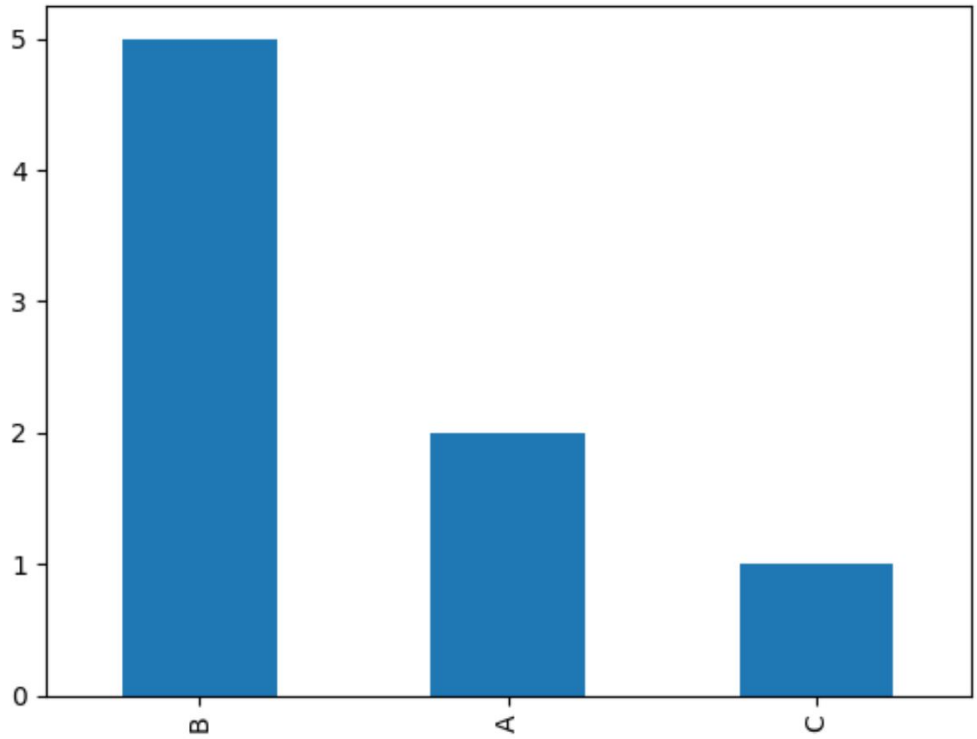
As observed in the output above, Team 'B' is the most frequent category with 5 occurrences, followed by 'A' with 2, and 'C' as the least frequent with 1 occurrence. This numerical distribution--B, A, C--will serve as the baseline for evaluating how each plotting method affects the final visualization order.

Practical Visualization: Plotting in Descending Order

Our first practical application utilizes the default behavior of the **value_counts()** output when combined with the .plot() method. Since **value_counts()** inherently sorts the resulting Series by count in descending order, this visualization is achieved with minimal chaining. This method is the standard and most intuitive way to generate a visualization that immediately emphasizes the majority categories.

When creating a frequency bar chart (by setting `kind='bar'`), this descending order ensures that the visual hierarchy aligns perfectly with the statistical importance of the categories, making it effortless for viewers to identify the most common elements in the dataset.

 # Plotting value counts of team in descending order (Default)
df.team.value_counts().plot(kind='bar')



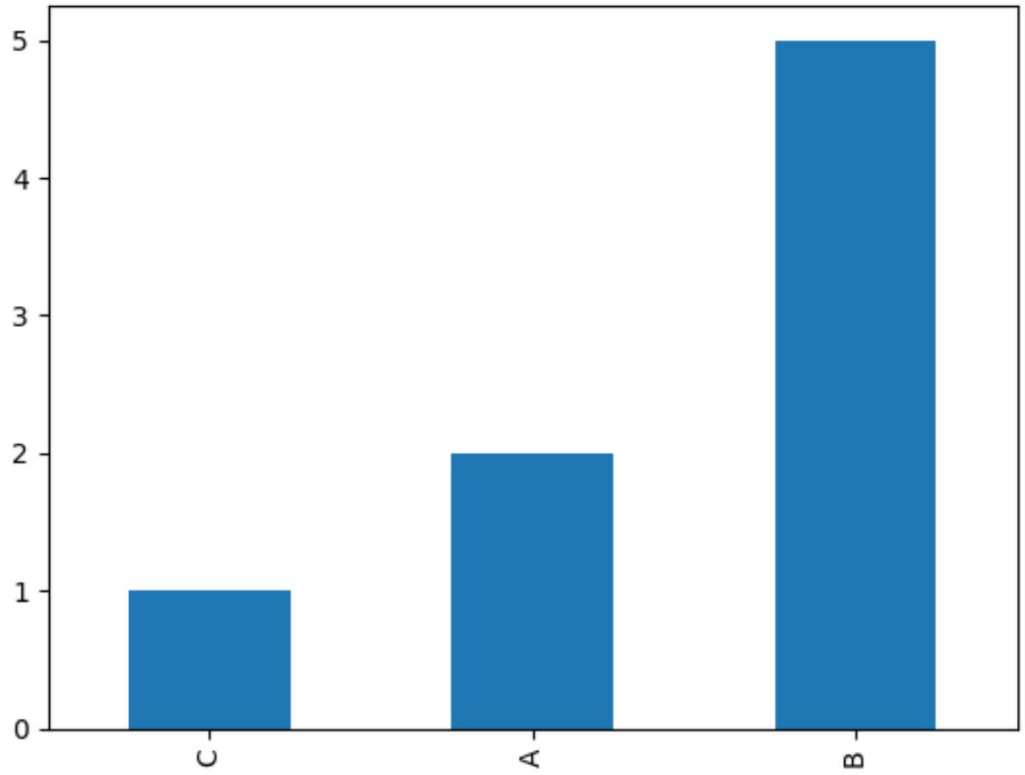
The resulting visual accurately reflects the frequency distribution: Team 'B' is displayed first with the highest bar, followed by 'A', and finally 'C'. The **x-axis** labels the categorical team names, while the **y-axis** accurately reflects the counts. A valuable alternative for presentations is the horizontal orientation, which can be achieved simply by changing the `kind` argument to **'barh'**.

Practical Visualization: Plotting in Ascending Order

In contrast to the default view, there are scenarios in data analysis where highlighting the least frequent occurrences is more critical than showing the dominant ones. To achieve this reverse sorting, we must explicitly call the **.sort_values()** method immediately after calculating the value counts. By default, this method sorts the values in ascending order.

This technique is invaluable when conducting anomaly detection or studying rare events, as it visually emphasizes the categories that might be overlooked in a standard descending frequency chart. It forces the viewer's attention toward the smaller, less represented segments of the data distribution.

 # Plotting value counts of team in ascending order
df.team.value_counts().sort_values().plot(kind='bar')



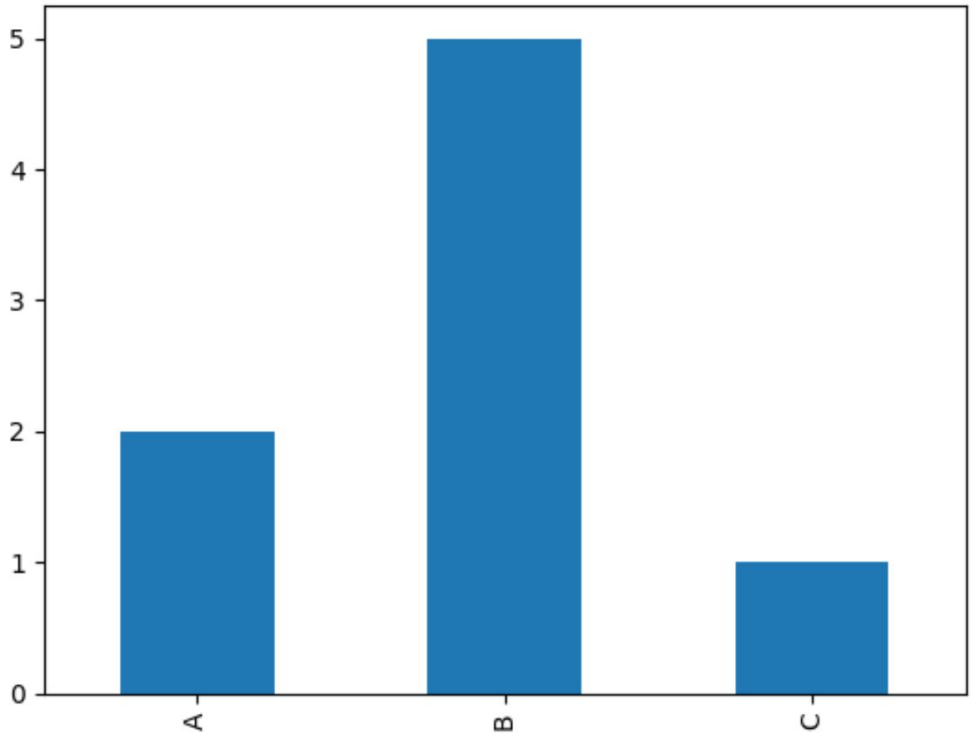
The visual output clearly demonstrates the reversal of the sequence. The bar chart now begins with 'C', the least frequent team, followed by 'A', and concludes with 'B', the most frequent. This deliberate sorting ensures that the data is presented from the lowest count to the highest, providing a distinct perspective on the distributional tail.

Practical Visualization: Plotting by Original Appearance Order

In some analyses, overriding frequency-based sorting is essential if the order in which the unique categories first appeared in the original DataFrame column has intrinsic meaning. For example, if categories represent stages in a process, preserving the order of appearance ensures chronological accuracy. Pandas facilitates this by allowing us to index the results of **value_counts()** using the sequence provided by the **.unique()** function.

The **.unique()** function retrieves the unique values in the order of their first appearance within the column. By passing this sequence back to index the frequency counts, we force the resulting Series--and consequently the plot--to adopt the original sequence before visualization.

 # Plotting value counts of team in order they appear in DataFrame
df.team.value_counts().plot(kind='bar')



Since 'A' appears first in the 'team' column, followed by 'B', and then 'C', the resulting bar chart is arranged in the sequence A, B, C, completely independent of their actual count (frequency). This specialized method is essential for maintaining context when category sequence is critical for accurate interpretation.

Summary and Resources for Further Pandas Exploration

Effective visualization of frequency distributions is a cornerstone of insightful data analysis. The Pandas library expertly combines data tabulation (via **value_counts()**) with integrated plotting capabilities, offering flexibility in how categorical data distributions are presented to stakeholders.

By mastering the use of descending (default), ascending (using **.sort_values()**), and original appearance order (using **.unique()**) methods, you gain precise control over the visual narrative of your data. This ability to choose the optimal visual sorting mechanism transforms raw counts into clear, actionable insights, greatly enhancing the communicative power of your analysis.

To further enhance your command of Pandas for complex data manipulation and visualization tasks, we recommend exploring these related tutorials:

How to Filter a Pandas DataFrame by Multiple Conditions
How to Group By Multiple Columns in Pandas
How to Calculate Rolling Average in Pandas

Introduction to Time Series Analysis with Pandas