Learning to Create Broken Axis Plots in R Using plotrix

Authored by

Mohammed looti

November 12, 2025

The Necessity of Broken Axis Plots in Data Visualization

In the realm of data visualization, effectively communicating complex information often requires specialized techniques. Occasionally, you may encounter datasets where certain data values are significantly separated from the main cluster, creating a situation where a standard plot becomes visually inefficient or misleading. Trying to display data points that span several orders of magnitude on a single continuous axis can compress the main body of the data, rendering subtle but important variations invisible.

To address this challenge, statisticians and data scientists frequently employ a technique known as a broken axis plot, also referred to as a discontinuity or segmented axis plot. This method allows you to visually compress the vast, empty space between two disparate ranges of values, ensuring that both the high-magnitude outliers and the densely clustered main data points are presented with appropriate resolution and clarity. While this technique must be used judiciously to avoid misrepresenting the data scale, it is an invaluable tool for improving readability in specific charting contexts.

Introducing the plotrix Package and the gap.plot() Function

For users working within the R environment, the simplest and most robust way to generate a broken axis plot is by utilizing the powerful plotrix package. The plotrix package is a dedicated library offering a wide array of specialized plotting functions and graphical devices that extend the capabilities of base R graphics. If you have not yet installed this package, you must execute `install.packages('plotrix')` in your R console before proceeding with the examples below.

The core function for creating broken axis plots within this package is **gap.plot()**. This function expertly handles the transformation of the coordinate system, allowing you to specify a range of values--a "gap"--that should be visually omitted from either the X or Y axis without losing the integrity of the remaining plotted data values. By default, **gap.plot()** is configured to apply breaks along the Y-axis, which is the most common requirement for outlier management, but it is easily configured to break the X-axis instead.

Understanding the Core Syntax and Parameters of gap.plot()

The **gap.plot()** function requires a minimal set of inputs to define the plot data and the required discontinuity. Understanding the parameters is essential for effective visualization. The function uses the following basic syntax, though it accepts numerous optional arguments for customization:

```
gap.plot(x, y, gap, gap.axis="y", ...)
```

The primary arguments that define the structure and location of the break are:

x: A mandatory vector containing the values to be plotted along the horizontal X-axis.
y: A mandatory vector containing the values to be plotted along the vertical Y-axis.
gap: This is a critical argument that defines the range of values to be excluded from the axis. It must be provided as a vector of two numeric values: `c(minimum_value_to_omit, maximum_value_to_omit)`. All data values falling within this range will be visually skipped, and the plot space will be compressed.
gap.axis: A character string that specifies which axis should contain the gap. Set to `"y"` by default, applying the break to the vertical axis. If you wish to break the horizontal axis, you must explicitly specify `gap.axis="x"`.

It is important to reiterate that while the default behavior of **gap.plot()** is to break the Y-axis, simply specifying **gap.axis="x"** allows you to apply the discontinuity to the horizontal dimension instead. This flexibility ensures that the function can accommodate various types of data visualization needs where the independent variable (X) might possess a large, empty interval.

Practical Application 1: Creating a Break on the X-Axis

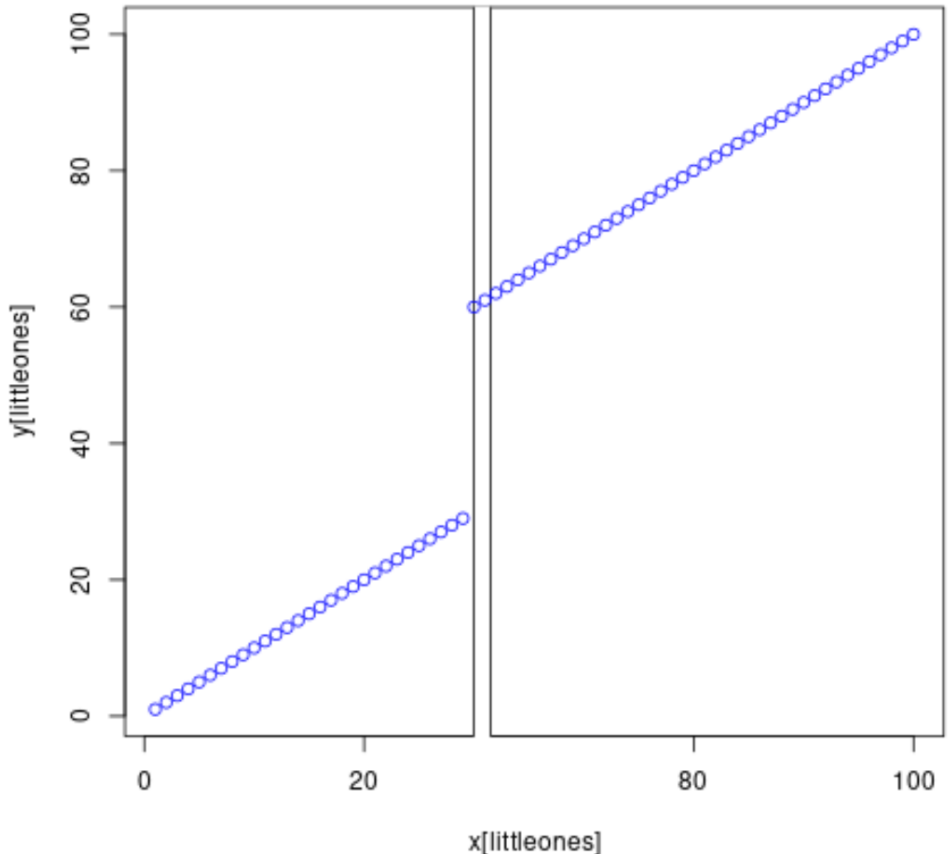
In our first practical example, we will demonstrate how to introduce a discontinuity on the horizontal X-axis. Imagine a scenario where we are plotting corresponding X and Y data values ranging from 1 to 100, but we know there is a significant, empty gap in the X-axis values between 30 and 60. Including this empty range would unnecessarily stretch the plot horizontally, making the data clusters at the beginning (1-30) and end (61-100) appear too compressed.

To address this, we use the **gap.plot()** function and specify the `gap` argument as `c(30, 60)`. Crucially, we must also set `gap.axis='x'` to ensure that the function applies this break horizontally. The following code snippet defines our sample vectors and executes the plotting command using the plotrix package:

library(plotrix)

```
#define x and y vectors
x <- 1:100
y <- 1:100

#create plot of x vs. y with gap on x-axis between 30 and 60
gap.plot(x, y, gap=c(30, 60), gap.axis='x', col='blue')
```



Upon reviewing the generated visualization, we can clearly observe the discontinuity marked on the X-axis, successfully omitting the range between 30 and 60, just as specified in our function call. This compression allows the remaining data to utilize the plot area more effectively. It is worth noting that if you encounter an error when executing the **gap.plot()** function, it is highly likely that the **plotrix** package has not been installed; remember to run `install.packages('plotrix')` first, followed by `library(plotrix)`.

Practical Application 2: Creating a Break on the Y-Axis

The most common use case for broken axis plots involves handling extreme outliers or large gaps in the dependent variable (Y-axis). For this demonstration, we will use the same input vectors (X and Y ranging from 1 to 100) but apply the gap vertically, again targeting the range between 30 and 60. Since breaking the Y-axis is the default behavior of the **gap.plot()** function, we can omit the `gap.axis='y'` argument, simplifying our code while achieving the desired result.

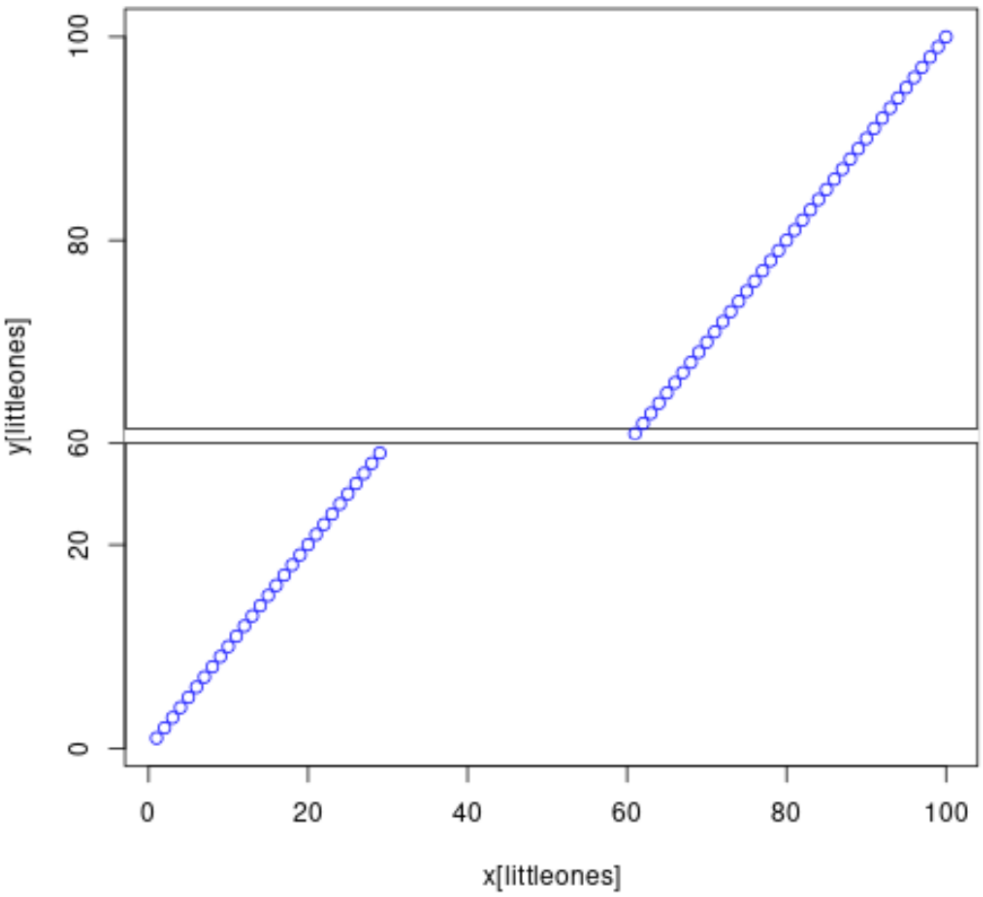
When visualizing data with a vertical gap, the function compresses the plot space where the data values are missing. This is particularly useful when comparing two groups of observations: one group with low values (e.g., 1-29) and another group with high values (e.g., 61-100), where the intermediate range contains no meaningful observations. The following syntax executes the Y-axis

break:

library(plotrix)

#define x and y vectors
x <- 1:100
y <- 1:100

#create plot of x vs. y with gap on y-axis between 30 and 60
gap.plot(x, y, gap=c(30, 60), col='blue')



As expected, the resulting graph displays a clear break on the Y-axis between the points of 30 and 60. It is crucial to internalize that specifying `gap.axis='y'` is technically optional here because **gap.plot()** defaults to the vertical axis. However, explicitly stating the axis can sometimes improve code clarity and documentation, especially when working with complex scripts.

Customizing Broken Axis Plots with Base R Arguments

While the **gap.plot()** function handles the complex axis manipulation, it retains compatibility with the standard graphical parameters used in R's base plotting system. This modularity is highly beneficial, allowing users to enhance the visual appeal and informational content of their broken axis plots using familiar arguments such as **main**, **xlab**, and **ylab**.

For instance, a plot without a title or proper axis labels can often confuse the reader regarding the nature of the visualized data values. By integrating these customization options, we can provide essential context. We will modify the previous Y-axis break example to include a descriptive title and meaningful axis labels, thereby significantly improving the overall quality of the data visualization.

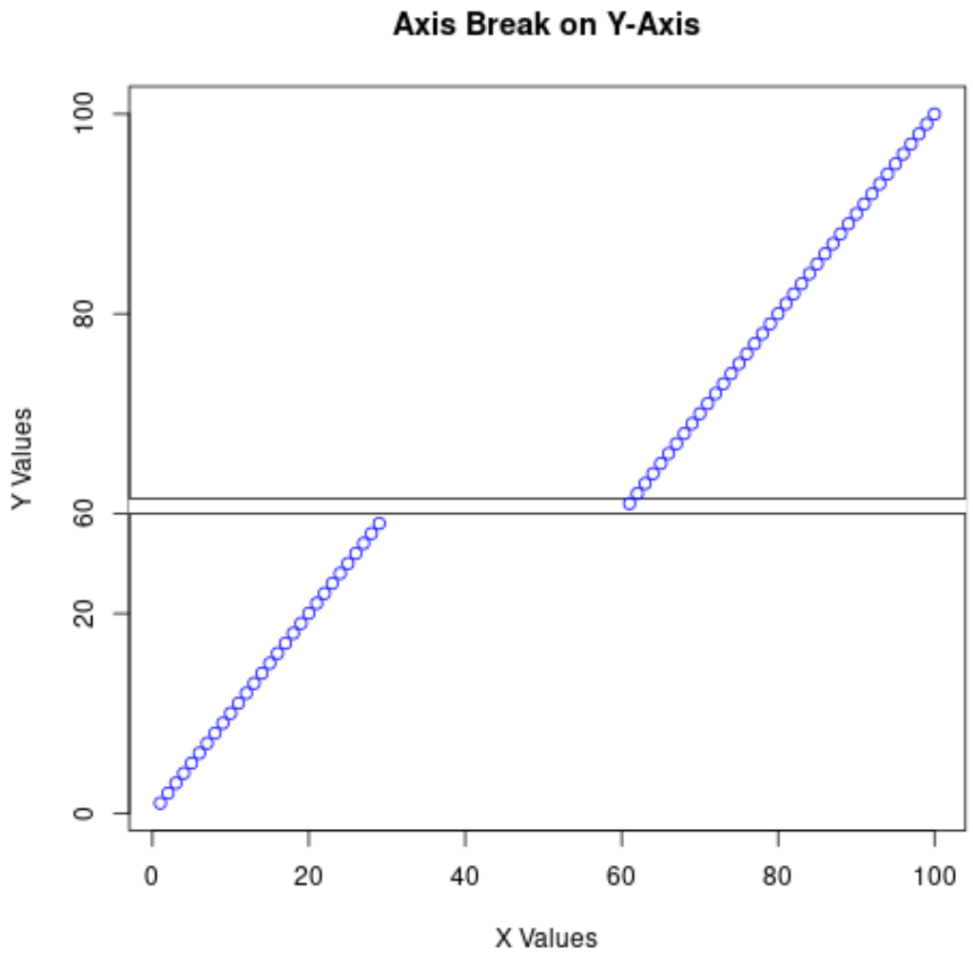
The following code snippet demonstrates how to pass standard base R plot arguments directly into the **gap.plot()** function:

```
library(plotrix)

#define x and y vectors
x <- 1:100
y <- 1:100

#create plot of x vs. y with gap on y-axis between 30 and 60
gap.plot(x, y, gap=c(30, 60), col='blue', main='Axis Break on Y-Axis',
xlab='X Values',
ylab='Y Values')
```

Executing this enhanced code produces a plot that is not only structurally sound with the axis break but is also clearly labeled, offering a professional and informative presentation:



Notice how the custom title and axis labels are now prominently featured. Users are encouraged to explore the extensive documentation related to base R plotting parameters (such as `xlim`, `ylim`, `pch`, and `cex`) and incorporate them into **gap.plot()** calls to achieve highly specific and customized graphical outputs tailored to their analytical requirements.

Additional Resources for R Plotting Mastery

Mastering specialized plotting techniques like broken axis plots significantly enhances your capacity for effective data visualization. The flexibility offered by packages like plotrix ensures that even unconventional data distributions can be represented clearly and accurately.

To continue building proficiency in R graphics, consider exploring tutorials on related common tasks. These skills often complement the ability to create customized axis plots:

How to Reverse Order of Axis in ggplot2