Learning to Visualize Data: Creating Strip Charts in R

Authored by

Mohammed looti

November 9, 2025

A **strip chart**, often referred to as a dot plot or dot strip plot, is a powerful visual tool utilized in statistical analysis to display the values of a **numeric vector** along a single axis. This visualization is particularly effective for assessing the underlying **distribution of data**. While similar in purpose to histograms or density plots, strip charts offer a clear view of every individual data point, making them invaluable when dealing with smaller sample sizes where summary statistics might obscure key details.

Unlike **boxplots**, which rely on quartiles and medians, the strip chart allows the viewer to see exactly where observations fall. This direct representation of raw data points is especially useful when sample sizes are minimal, as it prevents the aggregation that can sometimes mask interesting patterns or outliers. This tutorial focuses on how to leverage the base graphics package in **R** to create these informative plots using the built-in **stripchart()** function.

The Core Function: stripchart() Syntax in R

The **stripchart() function** is the primary tool within R for generating these visualizations. It is highly flexible, allowing users to control point separation, color, axis labels, and orientation. Understanding its key arguments is essential for producing customized and meaningful plots.

The fundamental syntax required to execute this function is structured as follows, with several optional arguments available for fine-tuning the output:

stripchart(x, method, jitter, main, xlab, ylab, col, pch, vertical, group.names)

Below is a detailed breakdown of the most critical parameters used within the **stripchart() function**:

x: This is the only mandatory argument. It requires a numeric vector or a list containing multiple numeric vectors intended for plotting.
method: Determines how points with identical values are handled. The default, "overplot," causes points to stack directly on top of one another. Alternatively, you can specify "jitter" to slightly offset the points horizontally (or vertically), or "stack" to arrange them in non-overlapping layers.
jitter: Applicable only when `method = "jitter"` is used, this argument specifies the magnitude of the random displacement (jittering) applied to the points to prevent overlap.
main: Defines the primary title displayed at the top of the chart.
xlab: Sets the descriptive label for the x-axis.
ylab: Sets the descriptive label for the y-axis.
col: Controls the color applied to the plotted data points.
pch: Allows customization of the plotting symbol (the shape of the points).
vertical: If set to `TRUE`, the plot will be drawn vertically, rather than the standard horizontal orientation.

group.names: Used when plotting multiple vectors; this provides the labels printed alongside the plot to identify the different groups.

Creating a Strip Chart for a Single Variable

To demonstrate the fundamental usage of **stripchart()**, we will utilize the renowned **iris dataset**, which is conveniently built into the R environment. This dataset provides measurements on 150 iris flowers, covering four key variables: Sepal Length, Sepal Width, Petal Length, and Petal Width, categorized by species.

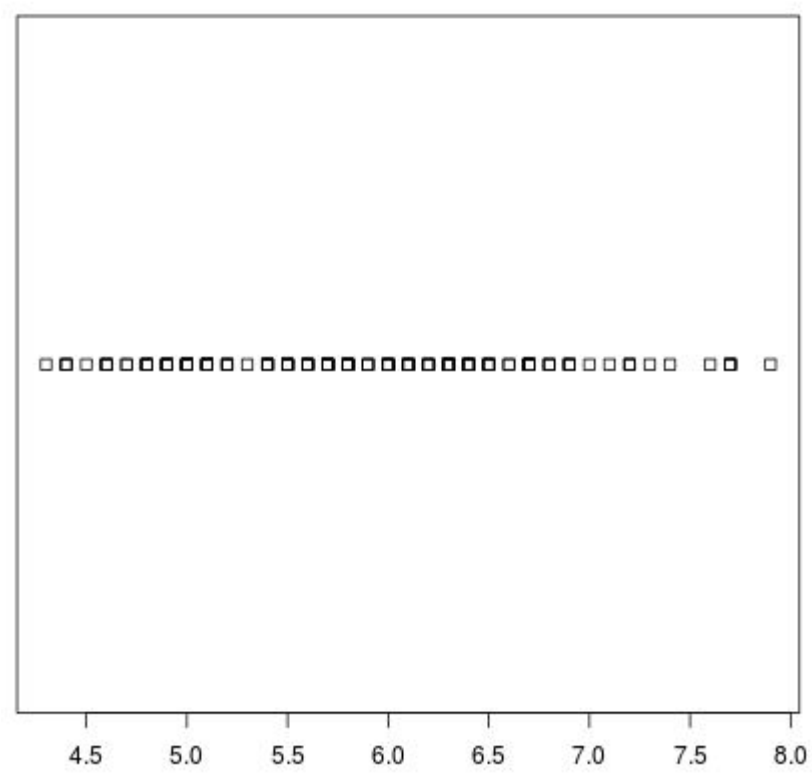
First, let us inspect the structure of the data by viewing the first few observations:

#view first six rows of *iris* dataset
head(iris)

Sepal.Length Sepal.Width Petal.Length Petal.Width Species
#1 5.1 3.5 1.4 0.2 setosa
#2 4.9 3.0 1.4 0.2 setosa
#3 4.7 3.2 1.3 0.2 setosa
#4 4.6 3.1 1.5 0.2 setosa
#5 5.0 3.6 1.4 0.2 setosa
#6 5.4 3.9 1.7 0.4 setosa

We will now create a rudimentary strip chart focusing solely on the distribution of the **Sepal.Length** variable. Since we provide no additional arguments, the plot defaults to horizontal orientation and uses the "overplot" method, which may result in numerous points hiding behind others if values are identical or very close.

stripchart(iris$Sepal.Length)

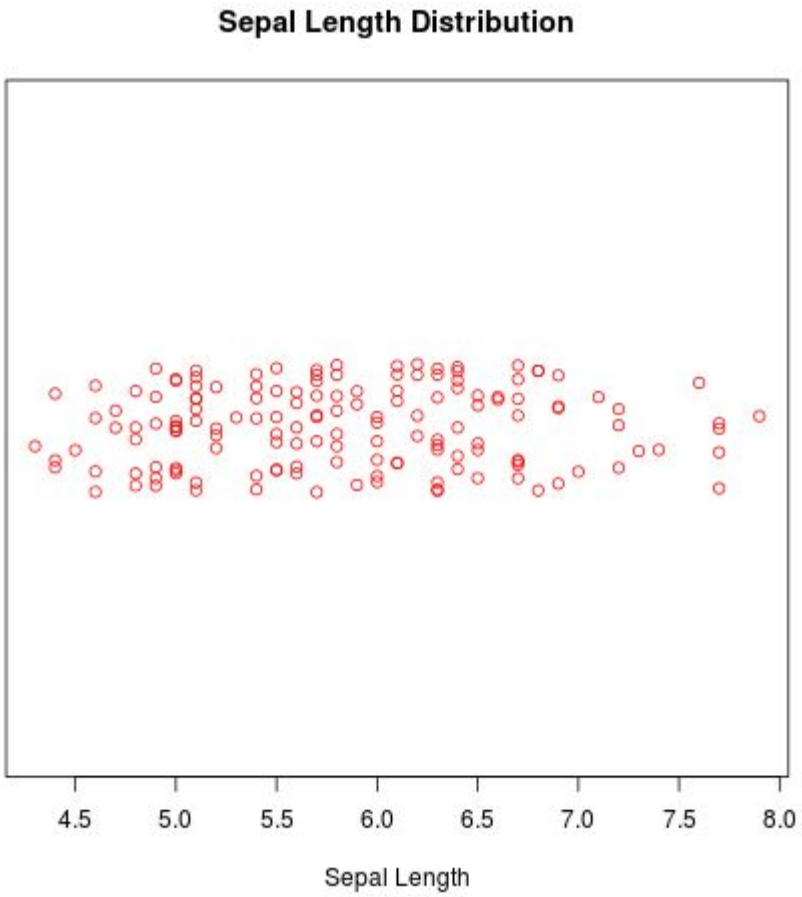


Visualizing Overlapping Data: Jitter vs. Stack

As seen in the basic plot above, the default "overplot" method makes it difficult to ascertain the true density of data points, particularly in areas where many measurements share the same value. To overcome this limitation, the **stripchart()** function provides two effective methods for separating overlapping points: `"jitter"` and `"stack"`.

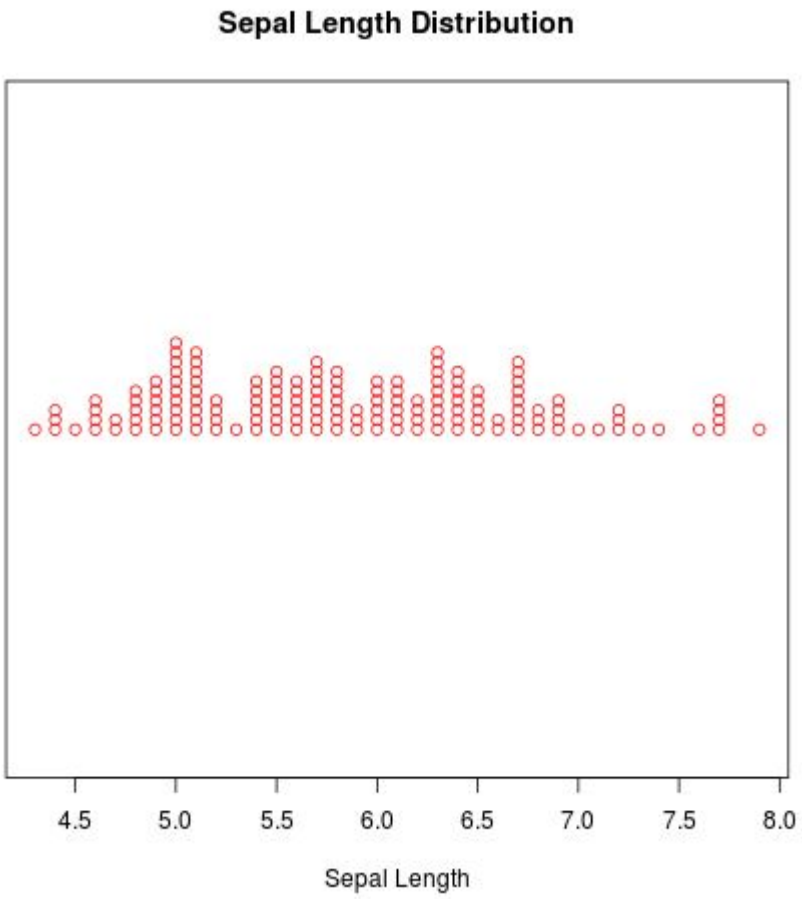
The `"jitter"` method introduces a small, random horizontal (or vertical) displacement to each point, allowing every individual observation to be visible. This is often the preferred method for viewing the exact count and spread of points while maintaining the integrity of the axis scale. We also apply several graphical parameters (`main`, `xlab`, `col`, `pch`) to enhance the chart's appearance and interpretability.

stripchart(iris$Sepal.Length,
main = 'Sepal Length Distribution',
xlab = 'Sepal Length',
col = 'red',
pch = 1,
method = 'jitter')



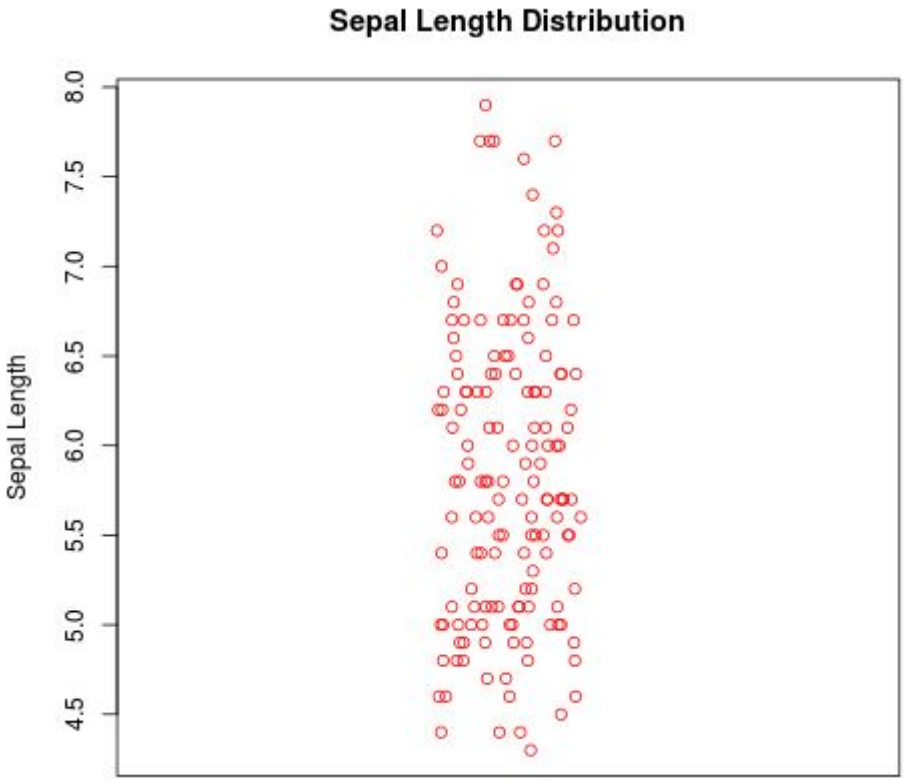
Alternatively, the `"stack"` method is used to arrange points with identical values into vertical stacks. This method clearly illustrates the frequency of specific values without introducing random noise, making it highly effective for data with discrete or rounded values. Notice how only the `method` argument needs to be changed to switch visualization styles:

stripchart(iris$Sepal.Length,
main = 'Sepal Length Distribution',
xlab = 'Sepal Length',
col = 'red',
pch = 1,
method = 'stack')



Finally, we can adjust the orientation using the `vertical = TRUE` argument. When plotting vertically, the primary measurement variable moves to the y-axis, requiring us to update the `ylab` argument instead of `xlab`. This presentation is often useful when comparing distributions side-by-side.

```
stripchart(iris$Sepal.Length,
main = 'Sepal Length Distribution',
ylab = 'Sepal Length',
col = 'red',
pch = 1,
method = 'jitter',
vertical = TRUE)
```



Plotting Multiple Variables Simultaneously

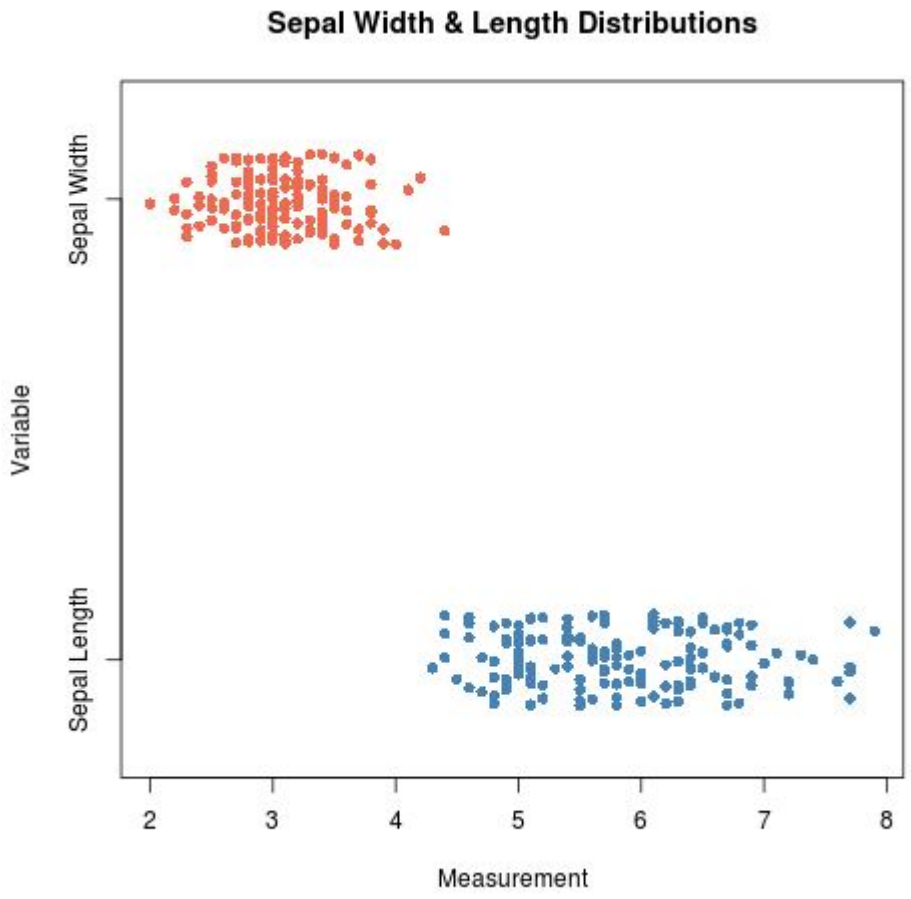
A major advantage of the strip chart is its ability to compare the distributions of several variables within a single visualization. To achieve this, we must pass a **list of numeric vectors** to the **stripchart()** function. The list structure allows R to treat each component as a distinct group to be plotted on separate lines.

In the following example, we create a list named `x` containing both the *Sepal Length* and *Sepal Width* variables from the **iris dataset**. We then use this list as the input for the function, applying different colors (`col`) for clarity and using the `jitter` method to visualize all 150 points for each variable.

#create list of variables
x <- list('Sepal Length' = iris$Sepal.Length, 'Sepal Width' = iris$Sepal.Width)

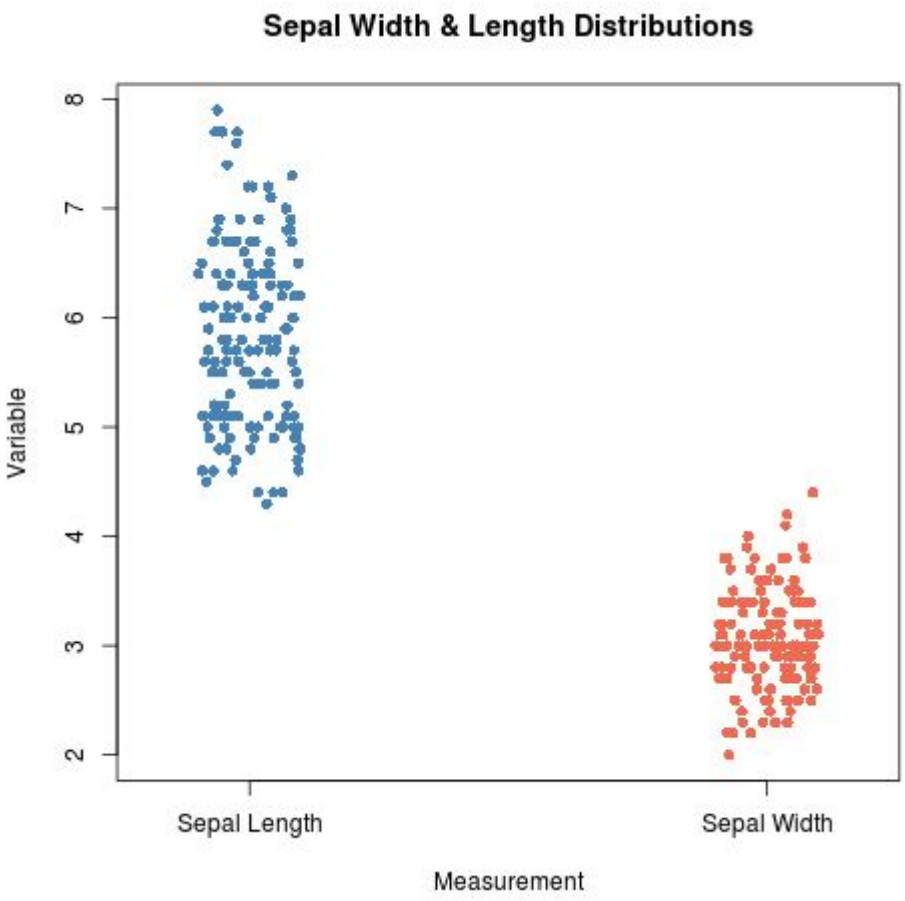
#create plot that contains one strip chart per variable
stripchart(x,
main = 'Sepal Width & Length Distributions',
xlab = 'Measurement',
ylab = 'Variable',

col = c('steelblue', 'coral2'),
pch = 16,
method = 'jitter')



Just as before, if the horizontal display is not ideal for your publication or analysis, these multiple distributions can easily be rotated to a vertical orientation by setting `vertical = TRUE`. This configuration is particularly effective when the group names are long or numerous.

stripchart(x, main = 'Sepal Width & Length Distributions',
xlab = 'Measurement',
ylab = 'Variable',
col = c('steelblue', 'coral2'),
pch = 16,
method = 'jitter',
vertical = TRUE)



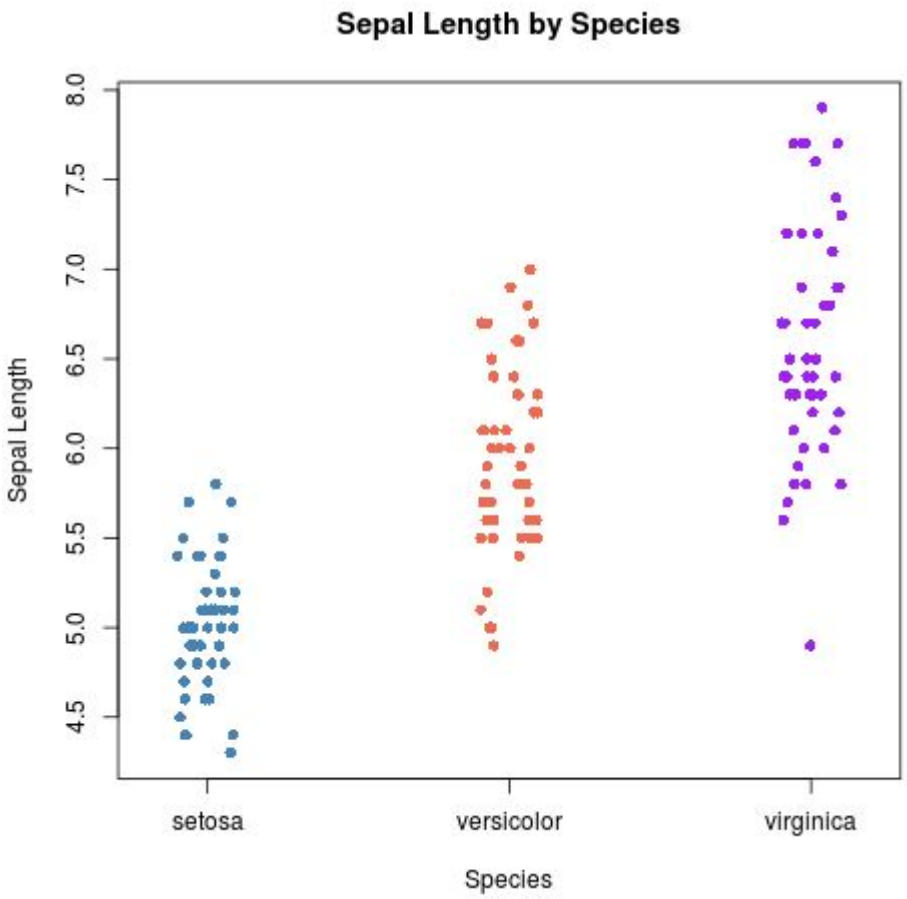
Grouping Data with Formula Notation

A highly concise method for plotting multiple distributions is using R's formula interface. This approach is ideal when you want to plot a numerical variable (the response, `y`) grouped by the categories of a factor variable (the predictor, `x`). The required format is `y ~ x`, which is standard across many R plotting functions.

Using the **iris dataset** again, we can examine the distribution of *Sepal Length* grouped by *Species*. Since *Species* has three distinct categorical values ("setosa", "versicolor", and "virginica"), R automatically creates three separate strip charts within the single plot area, one for each group.

stripchart(Sepal.Length ~ Species,
data = iris,
main = 'Sepal Length by Species',
xlab = 'Species',
ylab = 'Sepal Length',
col = c('steelblue', 'coral2', 'purple'),
pch = 16,

method = 'jitter',
vertical = TRUE)



This formula notation is extremely efficient for statistical exploration, allowing immediate visual comparison of how a continuous measurement varies across different experimental or natural groups. To delve deeper into the full capabilities and optional parameters of this function, you can access the official documentation directly within the R console:

?stripchart