

How to Easily Plot Mean Lines by Group in ggplot2

Authored by
stats writer

November 21, 2025

RECOMMENDED CITATION

stats writer (2025). *How to Easily Plot Mean Lines by Group in ggplot2*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=98704>

ggplot2 is the foundational R package for creating elegant and informative statistical graphics. When analyzing datasets, it is often critical to visualize not just the raw data points but also key summary statistics, such as the mean, segmented across different categorical variables or groups. Plotting a mean line specific to each group provides immediate visual context, allowing for quick comparisons between group performance or characteristics. This comprehensive guide details the precise methodology required to plot a horizontal mean line for distinct groups within a scatterplot using **R** and the **ggplot2** package.

The core technique involves two primary steps: first, calculating the required mean values externally using a robust data manipulation library like **dplyr**; and second, introducing these calculated values back into the visualization using a specialized geometric object. While methods like `stat_summary` exist for displaying aggregated statistics directly within the plot call, we will focus on the powerful and flexible approach of pre-calculating the means. This approach grants the user explicit control over the aggregation process and the visual representation of the mean lines, including aesthetics like color, line type, and thickness, ensuring that the final output is both accurate and visually appealing.

To achieve a clean, group-specific mean line, we leverage the `geom_hline` function in conjunction with a separate, aggregated data structure. This structure must contain the calculated mean for each group identified in the original dataset. By mapping the group variable to a color aesthetic and assigning the calculated mean to the `yintercept`, we can effectively overlay multiple distinct horizontal lines, each representing the average value for its respective category. The integration of data aggregation and visualization is key to generating these complex, multi-layered plots efficiently.

Prerequisites: Essential R Packages and Setup

Successfully implementing mean lines by group requires two fundamental packages from the **Tidyverse** ecosystem: **ggplot2** for the visualization layer and **dplyr** for the necessary data manipulation. These packages must be loaded into the R session before any data processing or plotting can occur. Using **dplyr** simplifies the complex task of splitting the original **data frame**, applying the mean calculation to each subset, and then recombining the results into a clean summary table ready for plotting.

The initial step in any analytical workflow should always be to ensure the required libraries are installed and attached. If you do not have these packages installed, you would typically use `install.packages("package_name")`. Once installed, the `library()` function makes the functions available for use. This ensures that the R environment recognizes specialized functions such as `group_by()`, `summarise()`, and the various `geom_` functions provided by the visualization package.

Once the packages are ready, the data structure itself must be prepared. The data needs to contain at least two numeric variables (for the x and y axes of the scatterplot) and one categorical variable (which will define the groups for which the mean is calculated). Our example uses a common scenario where individual observations (e.g., player statistics) are categorized by a grouping variable (e.g., 'team').

You can use the following basic syntax to plot a mean line by group in ggplot2, utilizing the aggregated data source:

```
#calculate mean points value by team
```

```
mean_team <- df %>% group_by(team) %>% summarise(mean_pts=mean(points))
```

```
#create scatterplot of assists vs points with mean line of points by team
```

```
ggplot(df, aes(x=assists, y=points)) +
```

```
geom_point(aes(color=team)) +
```

```
geom_hline(data=mean_team, aes(yintercept=mean_pts, col=team))
```

This particular example creates a scatterplot of the variables **assists** vs. **points**, then adds a line to represent the mean **points** value grouped by the **team** variable, demonstrating the seamless integration of pre-calculated statistics into the final visualization layer.

Data Aggregation: Calculating Group Means Using dplyr

The most crucial step before visualization is calculating the group means. We cannot simply tell **ggplot2** to draw a mean line for each group without first providing those specific mean values. This is where **dplyr** excels, utilizing a clear, sequential syntax facilitated by the pipe operator (`%>%`). This operator allows us to chain operations, making the code highly readable and expressive, detailing exactly how the data is transformed.

The aggregation process begins with the `group_by()` function, applied to the categorical variable-- in this case, `team`. This command logically splits the original **data frame** (`df`) into smaller chunks based on the unique values within the grouping column (Team A, Team B, Team C). Once grouped, subsequent functions operate independently on each of these subsets.

Next, the `summarise()` function is applied. This is the calculating engine; it reduces multiple rows within each group down to a single summary row. We instruct it to create a new column, `mean_pts`, which is computed as the `mean()` of the `points` variable for all observations belonging to the current group. The result of this operation is a new, concise **data frame**, `mean_team`, which contains only two columns: `team` and `mean_pts`. This aggregated **data frame** serves as the sole data source for plotting the mean lines.

The following example shows how to use this syntax in practice, beginning with the creation of the sample data.

Example: Plot Mean Line by Group in ggplot2

Suppose we have the following data frame in R that contains information about points and assists for basketball players on three different teams:

```
#create data frame
df <- data.frame(team=rep(c('A', 'B', 'C'), each=5),
  assists=c(2, 4, 4, 5, 6, 6, 7, 7,
  8, 9, 7, 8, 13, 14, 12),
  points=c(8, 8, 9, 9, 10, 9, 12, 13,
  14, 15, 14, 14, 16, 19, 22))
```

```
#view data frame
```

```
df
```

```
team assists points
```

```
1 A 2 8
```

```
2 A 4 8
```

```
3 A 4 9
```

```
4 A 5 9
```

```
5 A 6 10
```

```
6 B 6 9
```

```
7 B 7 12
```

```
8 B 7 13
```

```
9 B 8 14
```

```
10 B 9 15
```

```
11 C 7 14
```

```
12 C 8 14
```

```
13 C 13 16
```

```
14 C 14 19
```

```
15 C 12 22
```

Visualizing the Data: Creating the Scatterplot Foundation

The visualization process begins with the `ggplot()` function, which initializes the plot and defines the primary global aesthetics. In our case, the full **data frame**, `df`, is assigned as the main data

source. The aesthetic mappings (`aes()`) define `assists` for the x-axis and `points` for the y-axis. These mappings apply universally to all subsequent layers unless overridden.

The first geometric layer added is `geom_point()`, which is responsible for rendering the raw data points. Within this layer, we introduce a local aesthetic mapping: `aes(color=team)`. By mapping the categorical variable `team` to the color aesthetic, **ggplot2** automatically assigns a unique color to the points belonging to each team (A, B, and C). This step is essential for visual clarity and prepares the plot for the color-coded mean lines we will add next.

A key strength of the **ggplot2** grammar of graphics is its layering capability. The base layer defines the coordinates and raw data, while subsequent layers add complexity, such as statistical summaries, regression lines, or, in this case, horizontal markers. Maintaining the distinction between raw data visualization and summary visualization is paramount for creating a clean and informative chart.

Incorporating Group Means with `geom_hline`

The `geom_hline()` function is specifically designed to add horizontal lines to a plot. Unlike `geom_point()` or `geom_line()`, `geom_hline()` requires the fixed y-coordinate where the line must be drawn, specified by the `yintercept` aesthetic. Critically, because we want three distinct lines--one for each team--we must explicitly tell `geom_hline()` to use our aggregated data, `mean_team`, rather than the original data, `df`.

Within the `geom_hline()` call, we specify `data=mean_team`. Then, inside the aesthetic mapping, we set `yintercept=mean_pts`. Since the `mean_team` **data frame** contains one row for each team and the corresponding mean point value, **ggplot2** knows exactly where to place each line vertically. To ensure these lines are differentiated by group, we reuse the grouping variable by mapping `col=team` within the `aes()` of `geom_hline()`. This ensures that the color applied to the mean line matches the color assigned to the corresponding raw data points, completing the visual grouping.

This approach is fundamentally robust because it treats the mean lines as a separate data layer. If we had simply passed `data=df` to `geom_hline()`, the function would attempt to draw a line for every single observation in the original **data frame**, resulting in a nonsensical output or an error, as the `yintercept` would be calculated incorrectly. By isolating the calculated means into `mean_team`, we ensure that exactly three lines are drawn, one at the mean height for Team A, one for Team B, and one for Team C.

We can use the following complete code block to create a scatterplot of the variables **assists** vs. **points**, then add a line to represent the mean **points** value grouped by the **team** variable.

```
library(dplyr)
```

```
library(ggplot2)
```

```
#calculate mean points value by team
```

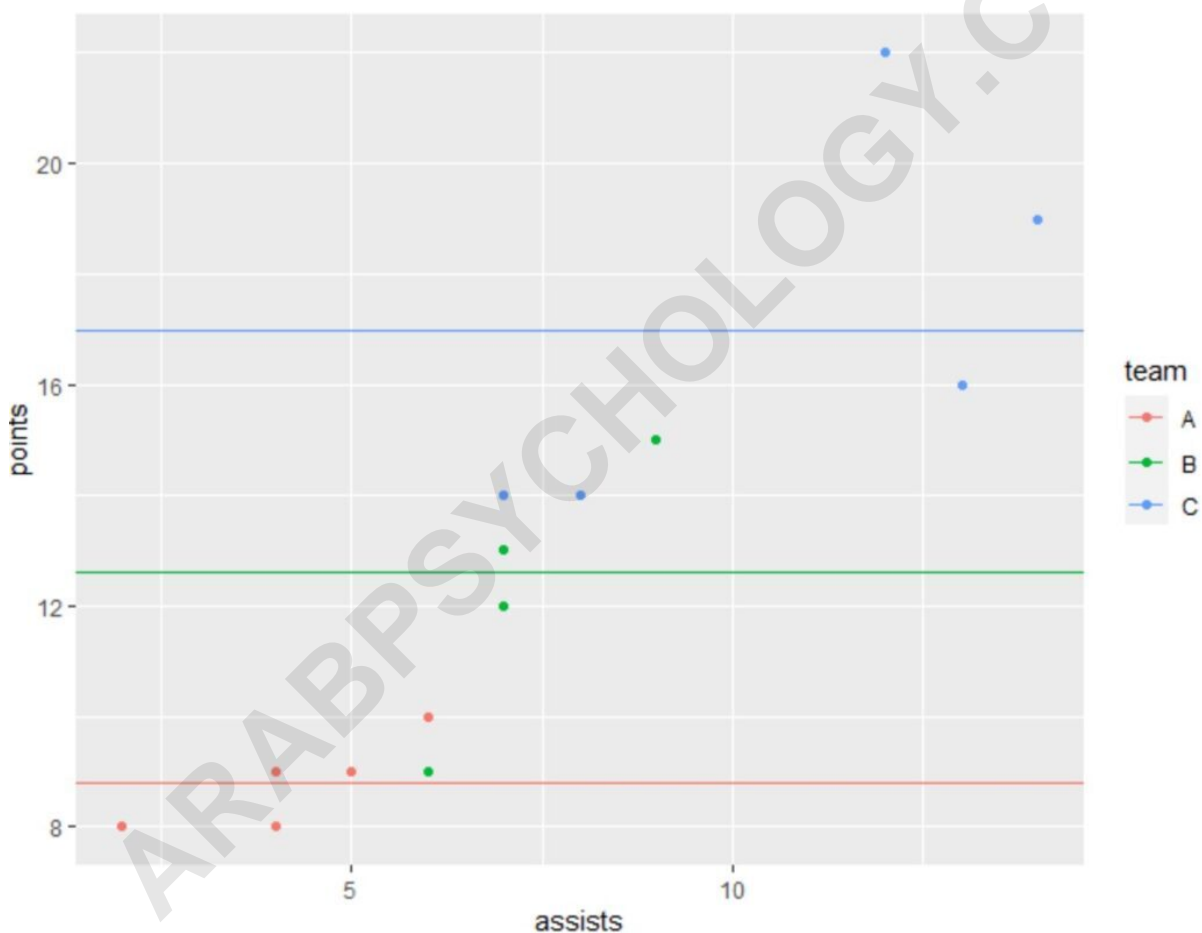
```
mean_team <- df %>% group_by(team) %>% summarise(mean_pts=mean(points))
```

```
#create scatterplot of assists vs points with mean line of points by team
```

```
ggplot(df, aes(x=assists, y=points)) +
```

```
geom_point(aes(color=team)) +
```

```
geom_hline(data=mean_team, aes(yintercept=mean_pts, col=team))
```



Analyzing the Calculated Means

Upon reviewing the generated plot, it is clear that the three horizontal lines correspond exactly to the average point production for each team. The lines are color-coded to display the mean points value for each team, allowing viewers to instantly correlate the central tendency of the data points with the summary line. Team C, which has the highest points, is represented by the highest

horizontal line, while Team A has the lowest mean line.

To confirm the visual representation, it is beneficial to examine the content of the summary **data frame**, `mean_team`, which we created using **dplyr**. This step validates the statistical computation that underpins the visualization, ensuring that the lines are accurately placed according to the statistical definition of the arithmetic mean.

Displaying the contents of `mean_team` provides the exact numerical values that were used as the `yintercept` coordinates for the `geom_hline()` layers. This transparency reinforces confidence in the graphic and provides precise data to accompany the visual insights derived from the scatterplot.

We can display the `mean_team` data frame we created to see the actual mean points values for each team:

#view mean points value by team

`mean_team`

```
`summarise()` ungrouping output (override with `.groups` argument)
```

```
# A tibble: 3 x 2
```

```
team mean_pts
```

```
1 A 8.8
```

```
2 B 12.6
```

```
3 C 17
```

From the output we can see the precise aggregated statistics:

The mean points value for players on team A is **8.8**.

The mean points value for players on team B is **12.6**.

The mean points value for players on team C is **17**.

These values perfectly match the locations of the lines on the y-axis of the scatterplot that we created, confirming the validity of the visualization technique.

Conclusion: Summarizing the Visualization Technique

Plotting group mean lines in **ggplot2** is an essential skill for any data analyst using **R**. By separating the task into data aggregation (using **dplyr**'s `group_by()` and `summarise()`) and visualization (using **ggplot2**'s layering system), we gain maximum control and flexibility over the final graphical output. This method ensures that statistical summaries are accurately represented

and clearly segregated by the categorical variables of interest.

The combination of `geom_point()` for raw data and `geom_hline` fed by an aggregated dataset is a powerful pattern. This pattern can be extended to calculate and display other statistics, such as medians, standard deviations, or confidence intervals, by simply adjusting the function used within `summarise()` and adding subsequent layers to the plot.

Ultimately, effective data visualization requires more than just showing data points; it demands the incorporation of context and key summary measures. The technique outlined here provides a robust, reproducible, and aesthetically pleasing way to highlight central tendencies across distinct groups within complex multivariate datasets.

ARABPSYCHOLOGY.COM