Learning to Visualize Data: Creating Lollipop Charts in R

Authored by

Mohammed looti

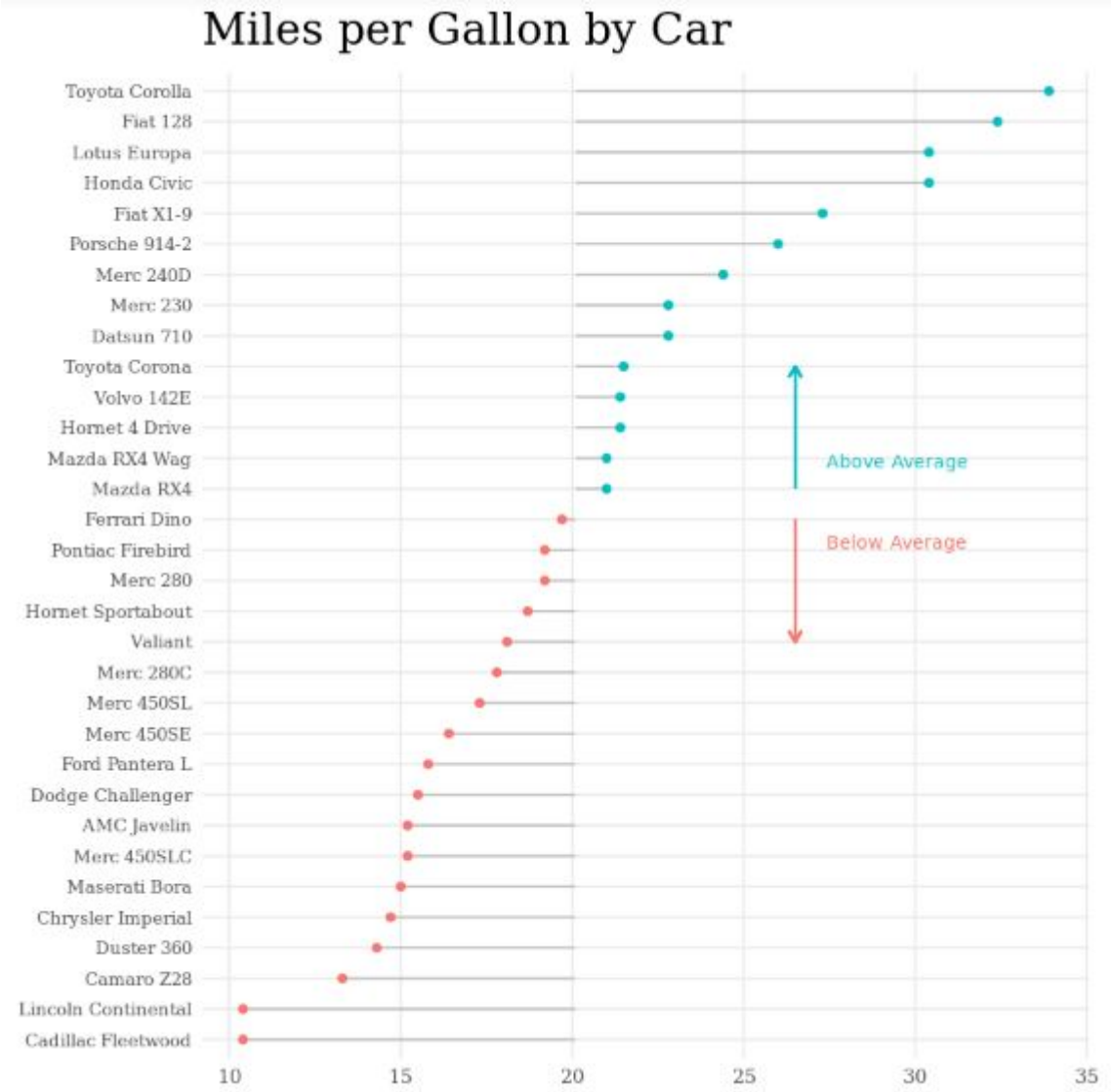
November 9, 2025

Understanding the Lollipop Chart: An Alternative to Bar Graphs

A lollipop chart represents a sophisticated and visually refined alternative to the traditional bar chart. Both chart types fulfill the essential data visualization requirement of comparing quantitative values across a categorical variable. However, unlike the area-heavy bars, the lollipop chart uses a thin line (the stick) connecting to a circle (the lollipop head) to denote magnitude. This design choice is particularly effective when visualizing large datasets or when seeking to minimize visual distraction.

The primary advantage of adopting the lollipop chart lies in its minimalist structure. By substituting the substantial visual weight of a bar with a slender line and a defined point, the chart significantly reduces "data ink," leading to a cleaner and less cluttered appearance. This technique is strategic, as it redirects the reader's focus away from the geometric properties of the bars and directly onto the precise quantitative value marked by the circle. Consequently, visual comparisons between categories become rapid and unambiguous, making this chart type increasingly popular for professional dashboards and analytical reports due to its inherent clarity and aesthetic appeal.

This tutorial provides a comprehensive guide for generating high-quality lollipop charts using the R programming language and the versatile ggplot2 package. We will cover all necessary steps, starting with data preparation, moving through basic plotting, and concluding with advanced customization techniques, including benchmarking against a mean value. Our goal is to replicate the professional visualization displayed below, showcasing the capabilities of R for aesthetic data visualization.



Preparing the Data: Utilizing the Built-in mtcars Dataset

For our practical demonstration, we will leverage the globally recognized, built-in R programming language dataset known as mtcars dataset. This dataset, derived from 1974 automotive statistics, provides rich quantitative detail across 32 distinct automobile models, focusing on variables like **mpg** (miles per gallon), horsepower, and cylinder count. It serves as an excellent case study for comparing a single metric across multiple named categories.

A key characteristic of the mtcars dataset, inherited from older R data standards, is that the identifying information for the car models is stored within the data frame's row names rather than in a standard column. For effective use with modern plotting libraries like ggplot2, which relies heavily on explicit column mappings, this structure must be modified. We must extract these row names and assign them to a dedicated categorical column before proceeding with the visualization layers.

Before any modifications, we inspect the initial structure of the dataset using the `head()` function to confirm the placement of the car names:

#view first six rows of *mtcars*
head(mtcars)

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	car
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4	Mazda RX4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4	Mazda RX4 Wag
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1	Datsun 710
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1	Hornet 4 Drive
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2	Hornet Sportabout
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1	Valiant

Constructing the Basic Lollipop Chart using ggplot2

The foundation of any sophisticated visualization in R is typically laid using ggplot2, which allows us to build graphics layer by layer. To create the lollipop chart, we require two fundamental geometric objects: a line segment for the stick and a point for the head.

Initially, we must prepare the data for ggplot2 by converting the car names (row names) into a proper column named `car`. This column will serve as our discrete y-axis variable. Following this crucial data transformation, we load the ggplot2 library, enabling access to its powerful functions.

The visualization itself is constructed by defining the aesthetic mappings (`aes`) within `ggplot()`, specifying `mpg` for the x-axis and `car` for the y-axis. We use `geom_segment` to draw the lines. For a standard plot starting at zero, the segment begins at `x = 0` and extends to `xend = mpg`, while the y-coordinates (`y` and `yend`) are consistently mapped to the `car` variable. The final component, the circle, is added using the `geom_point()` function, completing the basic lollipop chart structure.

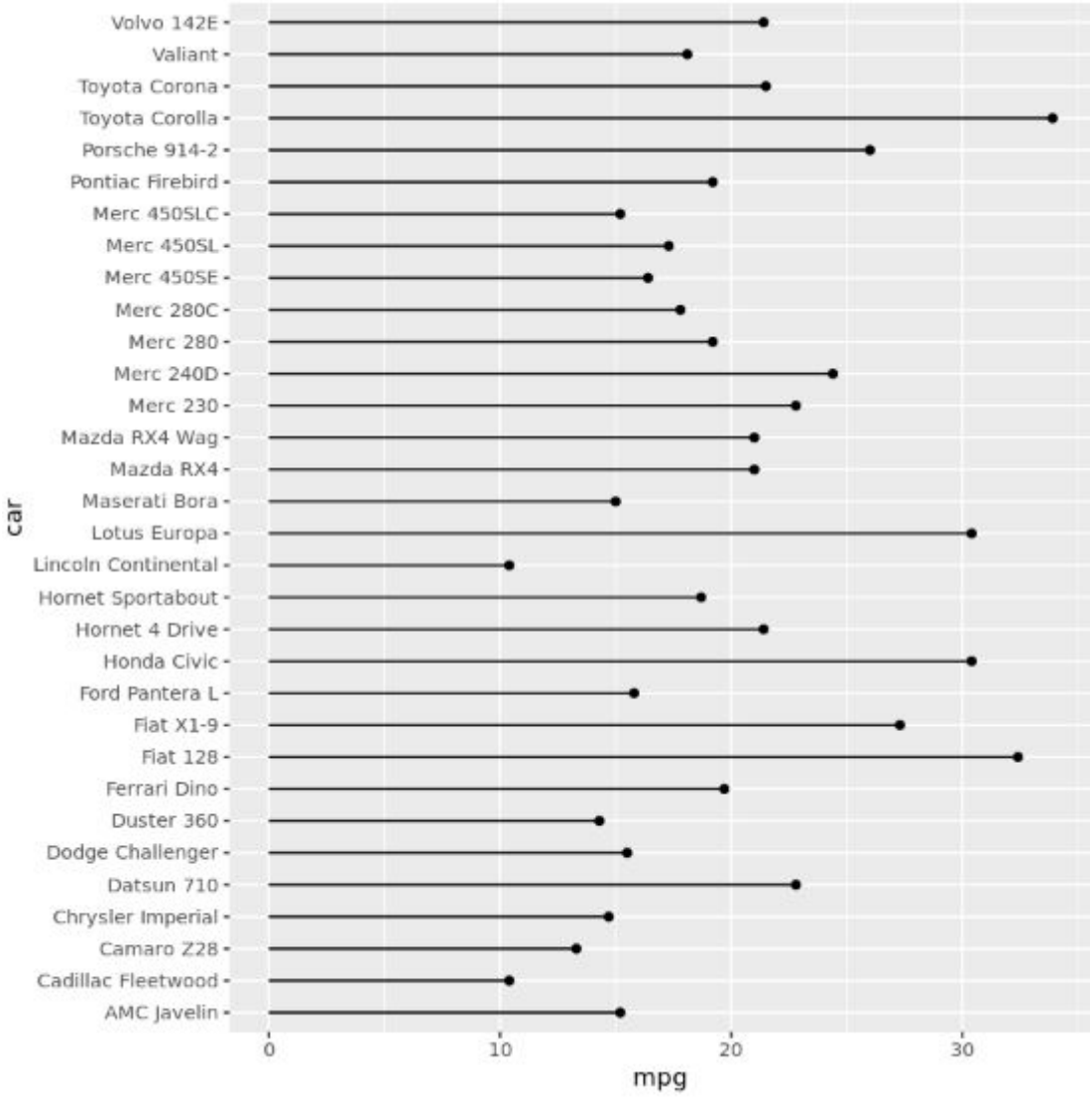
#create new column for car names
mtcars$car <- row.names(mtcars)

#load *ggplot2* library
library(ggplot2)

#create lollipop chart
ggplot(mtcars, aes(x = mpg, y = car)) +

```
geom_segment(aes(x = 0, y = car, xend = mpg, yend = car)) +
geom_point()
```

The output is a visually distinct comparison of the miles per gallon for each vehicle:



Implementing Data Labels and Custom Sizing

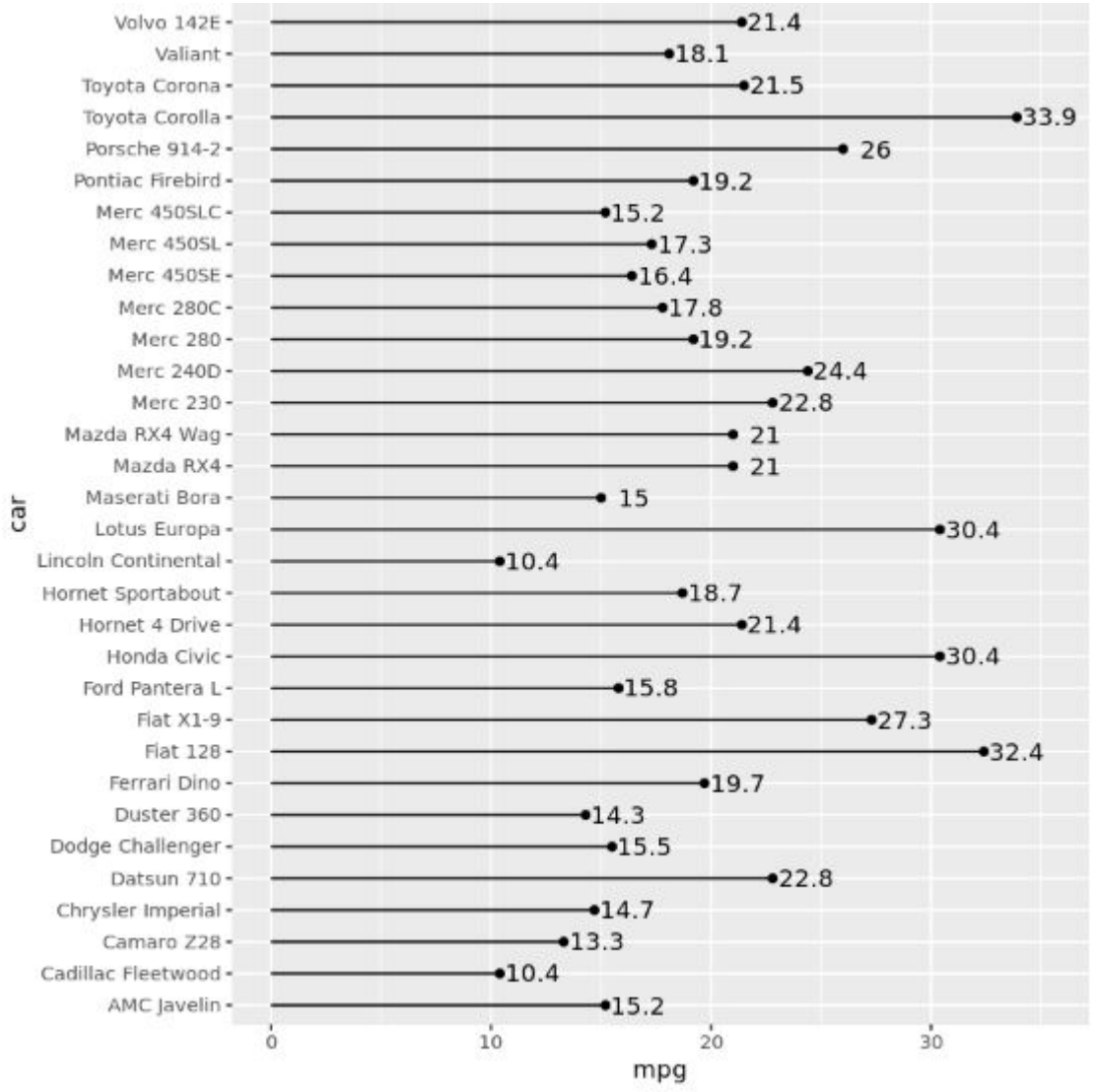
While the chart provides a clear visual ranking, precision is enhanced by adding explicit numerical labels. Relying solely on the axis scale introduces potential inaccuracies due to visual interpolation. We employ the `geom_text()` layer in ggplot2 to overlay these values directly onto the visualization.

To incorporate external labels, we map the `mpg` variable to the `label` aesthetic within the main plot call. Subsequently, `geom_text()` draws these values at the coordinates of the data points. To

prevent the labels from obscuring the points, we apply a slight horizontal offset using the `nudge_x` argument, which shifts the text outside the circle for maximum clarity.

```
ggplot(mtcars, aes(x = mpg, y = car, label = mpg)) +
geom_segment(aes(x = 0, y = car, xend = mpg, yend = car)) +
geom_point() +
geom_text(nudge_x = 1.5)
```

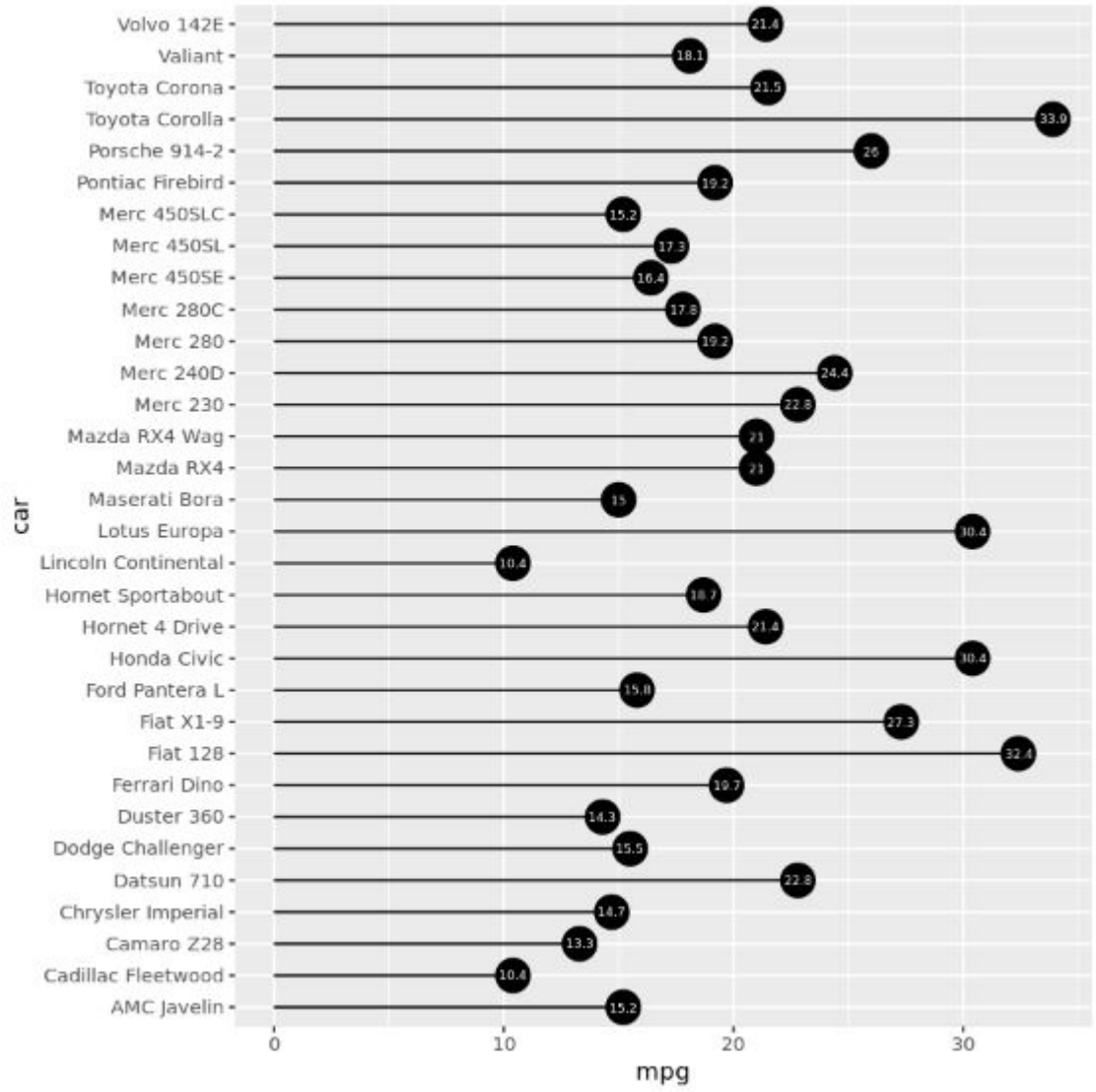
The resulting chart now clearly displays the exact mpg value alongside each point:



An alternative, highly stylized approach involves embedding the numerical labels directly within the circle heads. This requires modifications to the point geometry and the text aesthetics. We increase the point size substantially using the `size` argument in `geom_point()`. Then, we ensure legibility by changing the text color to contrast sharply with the point color--typically setting `color`

`= 'white'` within `geom_text()`. This creates a visually compact and modern graphic.

```
ggplot(mtcars, aes(x = mpg, y = car, label = mpg)) +
geom_segment(aes(x = 0, y = car, xend = mpg, yend = car)) +
geom_point(size = 7) +
geom_text(color = 'white', size = 2)
```



Benchmarking: Comparing Individual Values Against the Mean

One of the most powerful analytical uses of a lollipop chart is visualizing how individual data points deviate from a central reference point, such as the overall average. To achieve this, we must first calculate the mean of the `mpg` variable and then restructure the data to facilitate sorting and conditional coloring.

This data preparation task is efficiently handled by the dplyr package, an integral part of R's Tidyverse. We construct a data pipeline using the pipe operator (`%>%`) to perform several key manipulations on the `mtcars` data frame:

Ordering: We use the `arrange(mpg)` function to sort the cars by mileage, providing a logical flow for the final visual comparison.

Calculation: We create new columns using `mutate()`. `mean_mpg` stores the dataset's average mpg. A logical variable, `flag`, is created to identify whether an individual car's mpg is greater than the calculated average (TRUE/FALSE).

Factor Releveling: We explicitly redefine the `car` column as a factor, using the newly sorted order as the factor levels. This ensures ggplot2 maintains the desired vertical arrangement.

```
#load library dplyr
library(dplyr)

#find mean value of mpg and arrange cars in order by mpg descending
mtcars_new <- mtcars %>%
arrange(mpg) %>%
mutate(mean_mpg = mean(mpg),
flag = ifelse(mpg - mean_mpg > 0, TRUE, FALSE),
car = factor(car, levels = .$car))

#view first six rows of mtcars_new
head(mtcars_new)
```

The result of the dplyr pipeline confirms the new, sorted data structure, ready for advanced visualization:

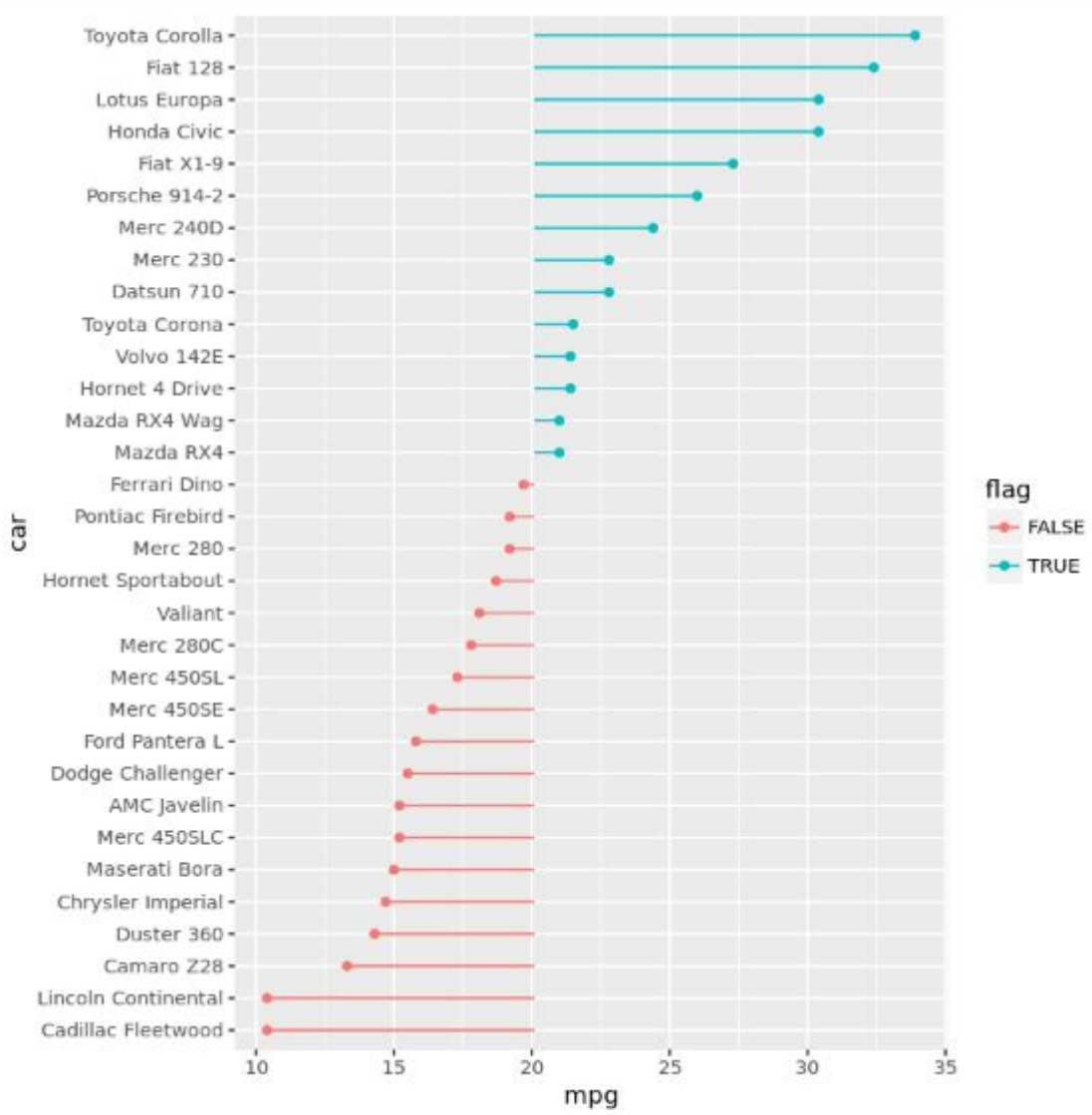
mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	car	mean_mpg	flag
10.4	8	472	205	2.93	5.250	17.98	0	0	3	4	Cadillac Fleetwood	20.09062	FALSE
10.4	8	460	215	3.00	5.424	17.82	0	0	3	4	Lincoln Continental	20.09062	FALSE
13.3	8	350	245	3.73	3.840	15.41	0	0	3	4	Camaro Z28	20.09062	FALSE
14.3	8	360	245	3.21	3.570	15.84	0	0	3	4	Duster 360	20.09062	FALSE
14.7	8	440	230	3.23	5.345	17.42	0	0	3	4	Chrysler Imperial	20.09062	FALSE
15.0	8	301	335	3.54	3.570	14.60	0	1	5	8	Maserati Bora	20.09062	FALSE

To plot the deviations, we make two critical changes to the ggplot2 code: first, the color aesthetic is

mapped to the new `flag` variable to visually distinguish above-average cars from below-average cars. Second, within `geom_segment`, we set the starting x-position (`x`) to `mean_mpg` instead of 0. The segments now originate at the average value, effectively illustrating the magnitude and direction of the deviation.

ggplot(mtcars_new, aes(x = mpg, y = car, color = flag)) +
geom_segment(aes(x = mean_mpg, y = car, xend = mpg, yend = car)) +
geom_point()

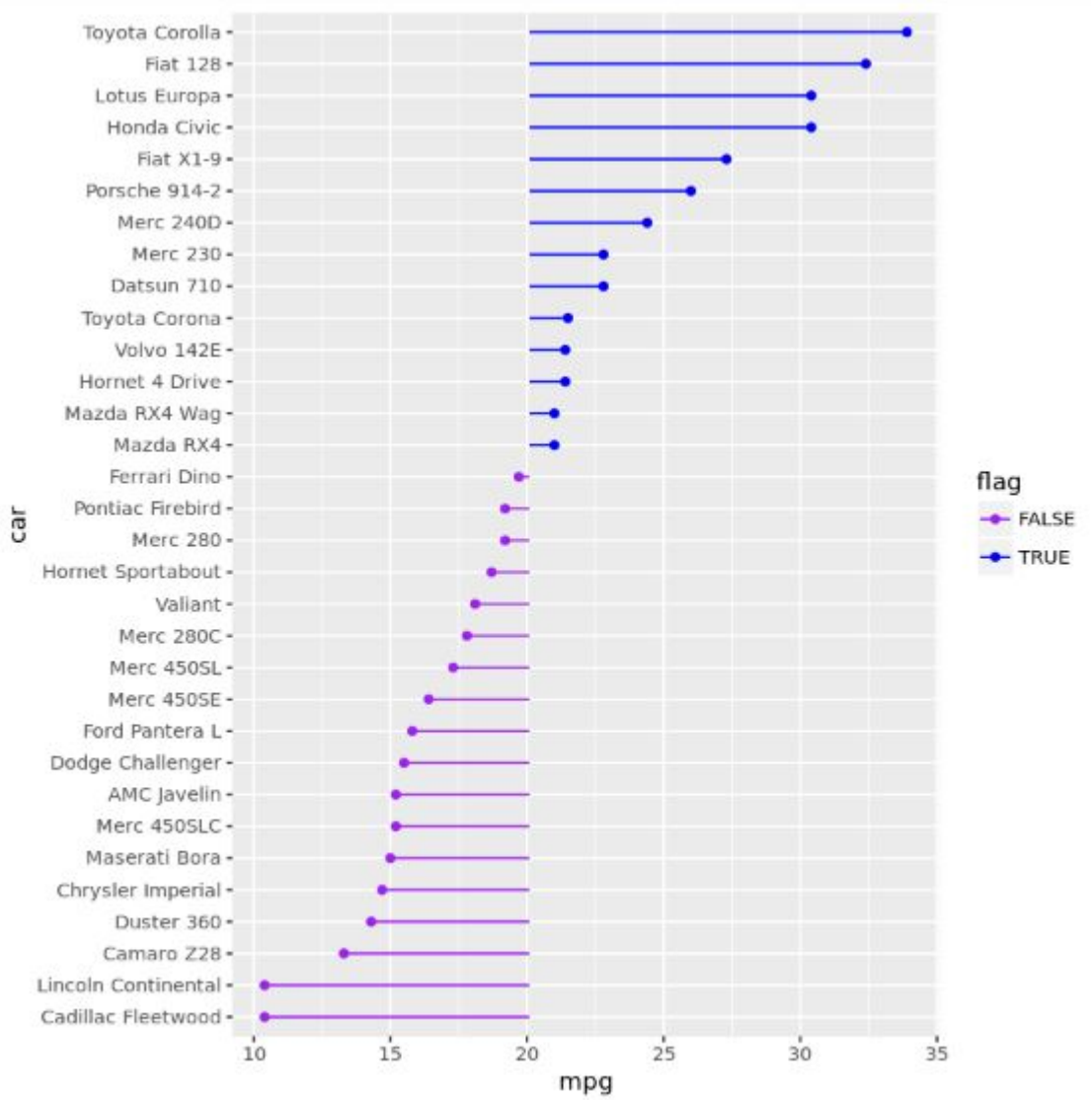
This chart provides an immediate comparative analysis:



While the default color scheme in R is functional, customizing colors is essential for aligning with specific design standards or improving visual contrast. We can manually assign specific colors to

the TRUE/FALSE values of the `flag` using the `scale_colour_manual` argument, ensuring our color choices are deliberate and effective.

```
ggplot(mtcars_new, aes(x = mpg, y = car, color = flag)) +
geom_segment(aes(x = mean_mpg, y = car, xend = mpg, yend = car)) +
geom_point() +
scale_colour_manual(values = c("purple", "blue"))
```



Refining Aesthetics for a Professional Finish

The final stage in creating a compelling visualization is applying advanced aesthetic controls available through ggplot2. This includes refining thematic elements, adding crucial annotations, and cleaning up unnecessary chart clutter to produce a polished, publication-ready graphic.

To finalize the chart, we implement several customizations:

Neutral Segments: We set the segment color to a neutral gray explicitly (outside of the aesthetic mapping) to ensure the focus remains on the colored points.

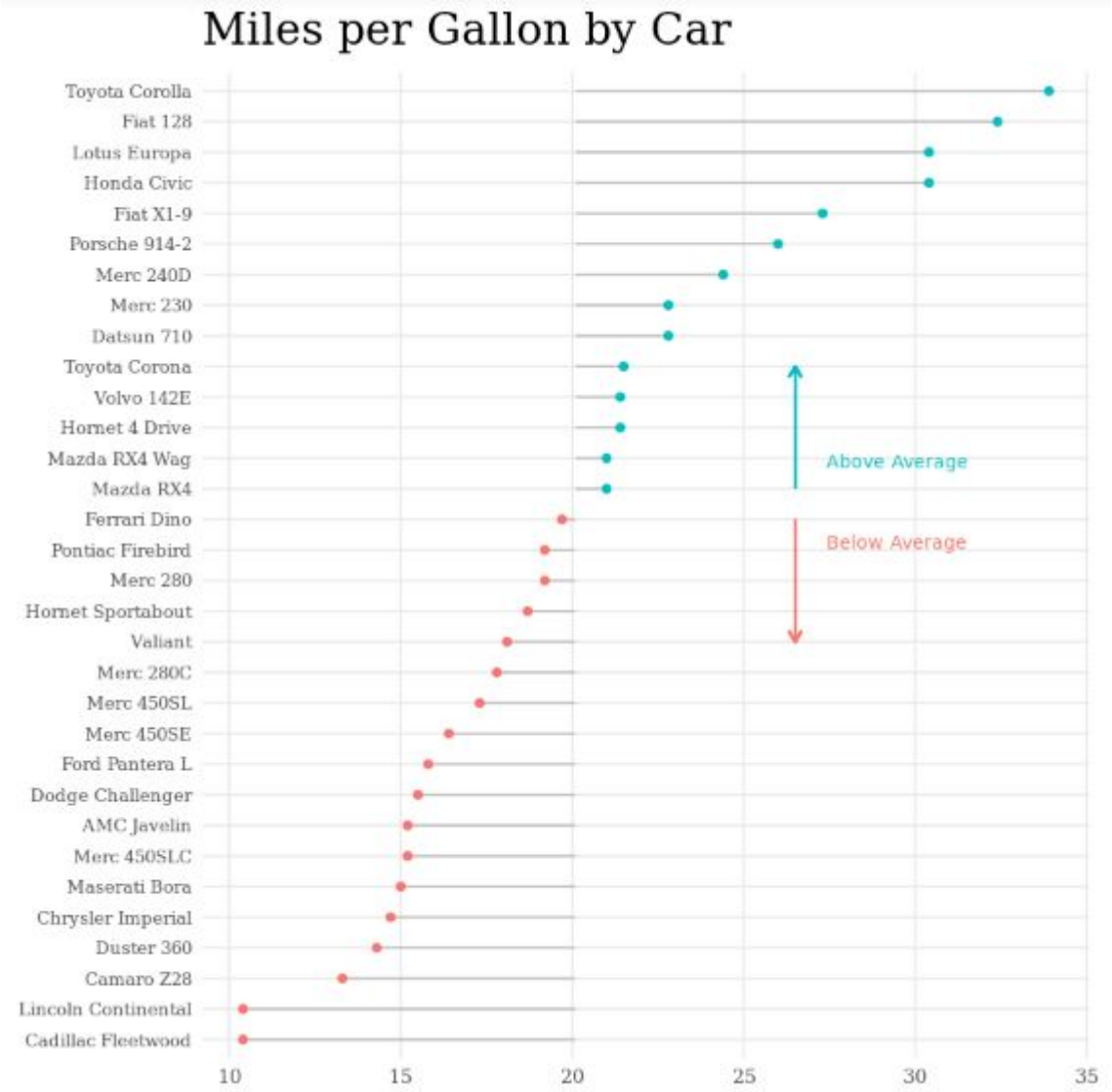
Annotations and Guidance: We use `annotate("text", ...)` and customized `geom_segment` with directional arrows to create a self-explanatory legend within the plot area, clearly linking the point colors to the concepts of "Above Average" and "Below Average."

Theming: We apply `theme_minimal()` and extensively customize the `theme()` arguments to remove extraneous gridlines, hide the redundant legend, define a professional font (e.g., "Georgia"), and format the plot title for maximum impact.

This comprehensive code block integrates all necessary layers for the final, refined visualization:

```
ggplot(mtcars_new, aes(x = mpg, y = car, color = flag)) +
geom_segment(aes(x = mean_mpg, y = car, xend = mpg, yend = car), color = "grey") +
geom_point() +
annotate("text", x = 27, y = 20, label = "Above Average", color = "#00BFC4", size = 3, hjust =
-0.1, vjust = .75) +
annotate("text", x = 27, y = 17, label = "Below Average", color = "#F8766D", size = 3, hjust =
-0.1, vjust = -.1) +
geom_segment(aes(x = 26.5, xend = 26.5, y = 19, yend = 23),
arrow = arrow(length = unit(0.2,"cm")), color = "#00BFC4") +
geom_segment(aes(x = 26.5, xend = 26.5 , y = 18, yend = 14),
arrow = arrow(length = unit(0.2,"cm")), color = "#F8766D") +
labs(title = "Miles per Gallon by Car") +
theme_minimal() +
theme(axis.title = element_blank(),
panel.grid.minor = element_blank(),
legend.position = "none",
text = element_text(family = "Georgia"),
axis.text.y = element_text(size = 8),
plot.title = element_text(size = 20, margin = margin(b = 10), hjust = 0),
plot.subtitle = element_text(size = 12, color = "darkslategrey", margin = margin(b = 25, l =
-25)),
plot.caption = element_text(size = 8, margin = margin(t = 10), color = "grey70", hjust = 0))
```

The final graphic represents a clear, aesthetic, and analytically robust comparison of the mtcars dataset against its central tendency:


Miles per Gallon by Car
Toyota Corolla
Fiat 128
Lotus Europa
Honda Civic
Fiat X1-9
Porsche 914-2
Merc 240D
Merc 230
Datsun 710
Toyota Corona
Volvo 142E
Hornet 4 Drive
Mazda RX4 Wag
Mazda RX4
Ferrari Dino
Pontiac Firebird
Merc 280
Hornet Sportabout
Valiant
Merc 280C
Merc 450SL
Merc 450SE
Ford Pantera L
Dodge Challenger
AMC Javelin
Merc 450SLC
Maserati Bora
Chrysler Imperial
Duster 360
Camaro Z28
Lincoln Continental
Cadillac Fleetwood
Above Average
Below Average
10
15
20
25
30
35