# Learn to Visualize Population Demographics: A Step-by-Step Guide to Creating Population Pyramids in R

Authored by

**Mohammed looti**

November 9, 2025

A population pyramid is a fundamental graphical tool used in demographic data analysis. It provides an immediate and comprehensive visual representation of the age and sex distribution within a given population. This specialized bar chart is not merely a statistical summary; it is a powerful indicator that helps analysts understand the current structure of a population, reflect historical events (like wars or baby booms), and forecast future population trends, including demands on healthcare, education, and social security systems.

The structure of the population pyramid reveals crucial insights into population dynamics. For instance, if the chart exhibits a rectangular or column-like shape, it signals a stationary population pattern. This pattern suggests that birth rates and death rates are relatively low and balanced, meaning older generations are being replaced by new generations of approximately the same size. This structure is common in developed nations with slow or zero population growth.

Conversely, a true pyramid shape, where the base is wide and the top is narrow, indicates an expansive population. This is characteristic of high birth rates and higher mortality rates in older age cohorts, leading to rapid population growth. Understanding these shapes--whether expansive, constrictive (inverted pyramid), or stationary--is essential for accurate demographic planning and policy implementation.

Within the chart's architecture, the visualization is horizontally split by gender, typically showing males on the left side and females on the right. Age groups are displayed vertically along the y-axis, usually in five-year increments, ascending from the bottom. The horizontal x-axis represents the percentage or total amount of the population within each age-sex category. This tutorial focuses on how to leverage the statistical programming language R, specifically using the powerful visualization package ggplot2, to construct these informative population pyramids.

## Preparing Demographic Data in R

Before plotting, the data must be organized into a suitable format. A population pyramid requires three primary variables: age cohorts, gender indicators, and the corresponding population count or percentage for each cohort. For visualization purposes in R, it is standard practice to store this information in a data frame structure, ensuring that each row corresponds to a specific combination of age and gender.

We will begin by creating a synthetic dataset that mimics real-world population data, spanning ages 1 to 100 and divided into male and female categories. We utilize the `set.seed(1)` command to ensure that this example remains perfectly reproducible across different environments, which is a best practice in statistical reporting. The population variable is then calculated using a mathematical transformation (inverse square root of age) and randomized noise (`runif`) to simulate a more realistic, decreasing population trend as age increases.

The final step in data preparation involves converting the raw population counts into percentages, relative to the total population. This normalization step is often crucial for population pyramids, as percentages provide a standardized scale for comparison across different population sizes. The following code demonstrates the creation and structure of this preparatory dataset:

```
#make this example reproducible
set.seed(1)

#create data frame
data <- data.frame(age = rep(1:100, 2), gender = rep(c("M", "F"), each = 100))

#add population variable
data$population <- 1/sqrt(data$age) * runif(200, 10000, 15000)

#convert population variable to percentage
data$population <- data$population / sum(data$population) * 100

#view first six rows of dataset
head(data)

# age gender population
#1 1 M 2.424362
#2 2 M 1.794957
#3 3 M 1.589594
#4 4 M 1.556063
#5 5 M 1.053662
#6 6 M 1.266231
```

## Constructing the Basic Population Pyramid with ggplot2

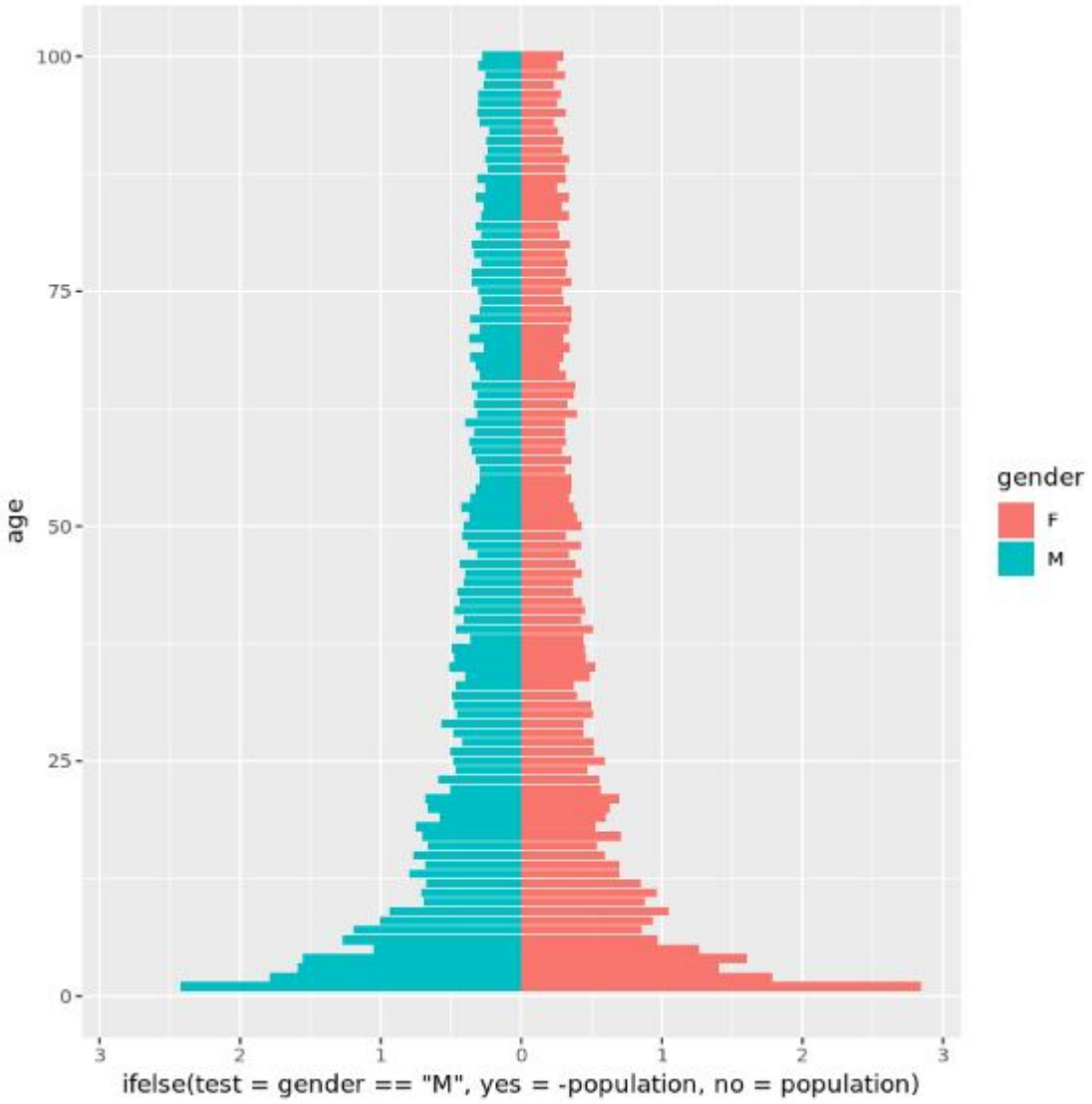
The power of the ggplot2 library lies in its layer-based grammar of graphics, which allows complex visualizations to be built up sequentially. To create the characteristic mirrored bar structure of a population pyramid, we must manipulate the data for one gender (typically male) to display on the negative side of the x-axis. This is achieved within the aesthetic mapping (`aes()`) argument using an `ifelse` conditional statement.

The core plotting process involves mapping the age variable to the x-axis and the conditionally signed population variable to the y-axis. The `fill` aesthetic is mapped to the gender variable to differentiate the male and female bars. We use the geom_bar function with `stat = "identity"` because we are supplying the actual bar heights (the population percentages) rather than counting observations.

Crucially, two additional functions are required to finalize the pyramid shape: `scale_y_continuous(labels = abs, limits = max(data$population) * c(-1,1))` ensures that the population percentages are displayed as positive values despite the negative mapping, and that the axis limits are symmetrical. Finally, `coord_flip()` rotates the entire chart 90 degrees, turning the horizontal bars into the vertical structure characteristic of a population pyramid, with age on the vertical axis.

**#load *ggplot2***
**library(ggplot2)**

#create population pyramid
ggplot(data, aes(x = age, fill = gender,
y = ifelse(test = gender == "M",
yes = -population, no = population))) +
geom_bar(stat = "identity") +
scale_y_continuous(labels = abs, limits = max(data$population) * c(-1,1)) +
coord_flip()




## Enhancing Visualization: Labels, Titles, and Colors

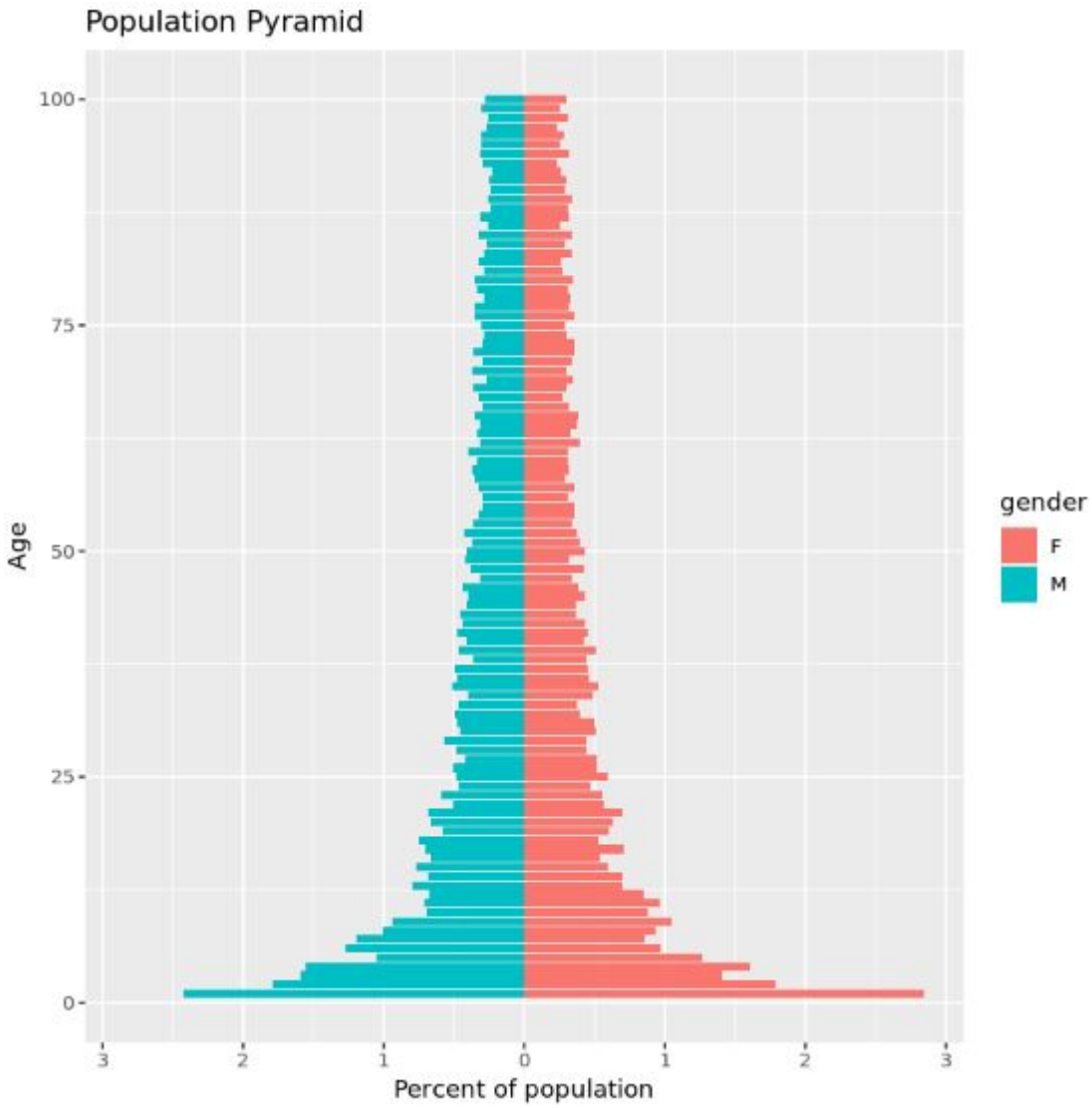
While the basic pyramid provides the structure, effective data visualization requires clear context provided through descriptive labels, titles, and appropriate color choices. The ggplot2 function `labs()` is the primary tool for adding a main chart title, specifying the labels for the x and y axes, and customizing the legend title, thereby drastically improving the chart's interpretability for any audience reviewing this demographic data.

In demographic visualization, color selection is crucial, often requiring adherence to specific conventions (e.g., blue for male, pink/red for female) or institutional branding guidelines. The default colors provided by ggplot2 are suitable but rarely customized. To manually define the colors used to fill the bars, we employ the scale_colour_manual function. This function allows us to map specific color values (e.g., "pink" and "steelblue") to the discrete gender categories defined in our R

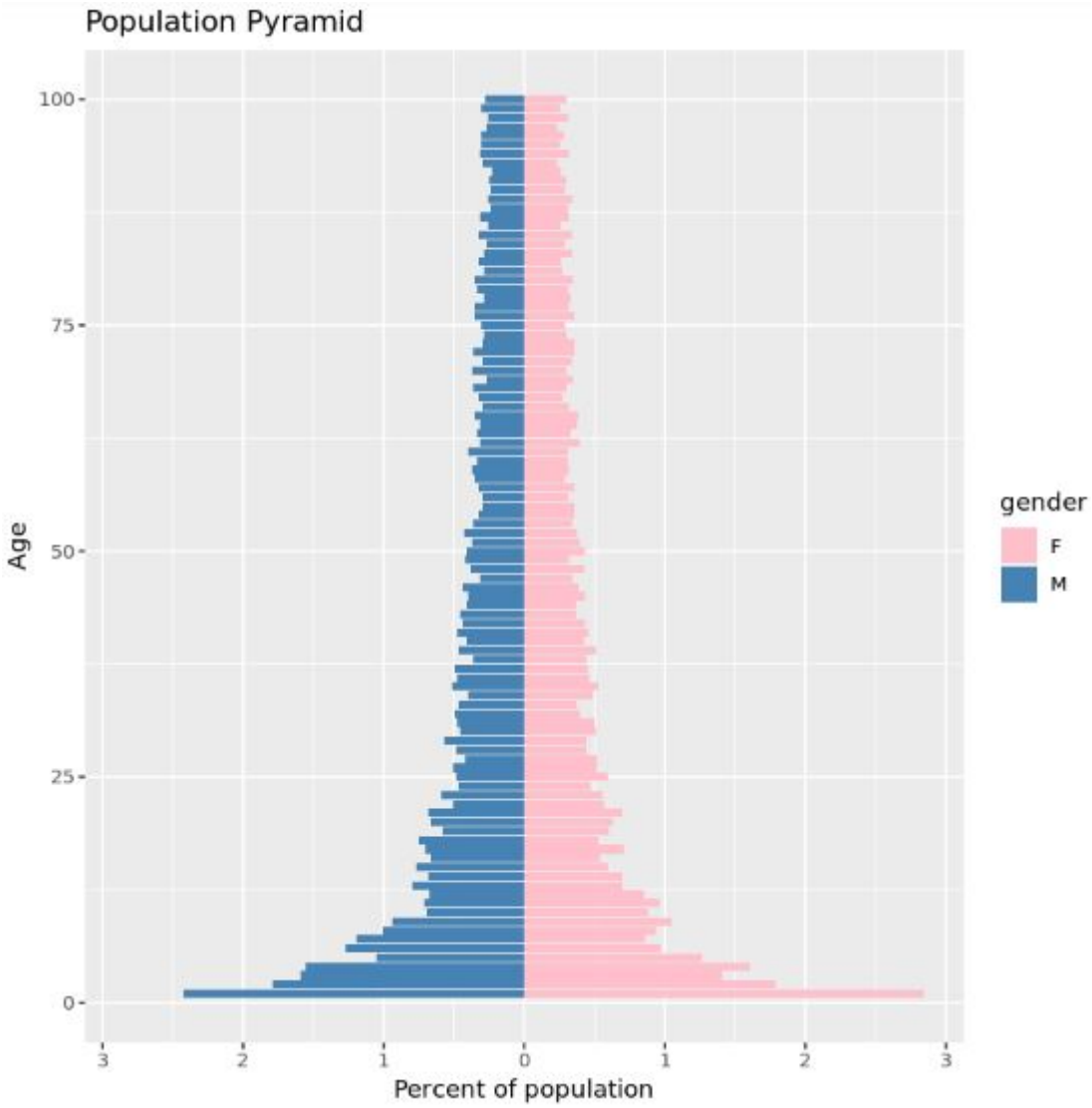
data frame.

A key parameter within `scale_colour_manual()` is `aesthetics = c("colour", "fill")`. By specifying both "colour" and "fill," we ensure that the defined colors are used both for the bar fill and, if applicable, for any associated colored lines or points, guaranteeing thematic consistency across the visualization. The following code blocks demonstrate the inclusion of titles, labels, and custom colors:

```
ggplot(data, aes(x = age, fill = gender,
y = ifelse(test = gender == "M",
yes = -population, no = population))) +
geom_bar(stat = "identity") +
scale_y_continuous(labels = abs, limits = max(data$population) * c(-1,1)) +
labs(title = "Population Pyramid", x = "Age", y = "Percent of population") +
coord_flip()
```




```
ggplot(data, aes(x = age, fill = gender,
y = ifelse(test = gender == "M",
yes = -population, no = population))) +
geom_bar(stat = "identity") +
scale_y_continuous(labels = abs, limits = max(data$population) * c(-1,1)) +
labs(title = "Population Pyramid", x = "Age", y = "Percent of population") +
scale_colour_manual(values = c("pink", "steelblue"),
aesthetics = c("colour", "fill")) +
coord_flip()
```



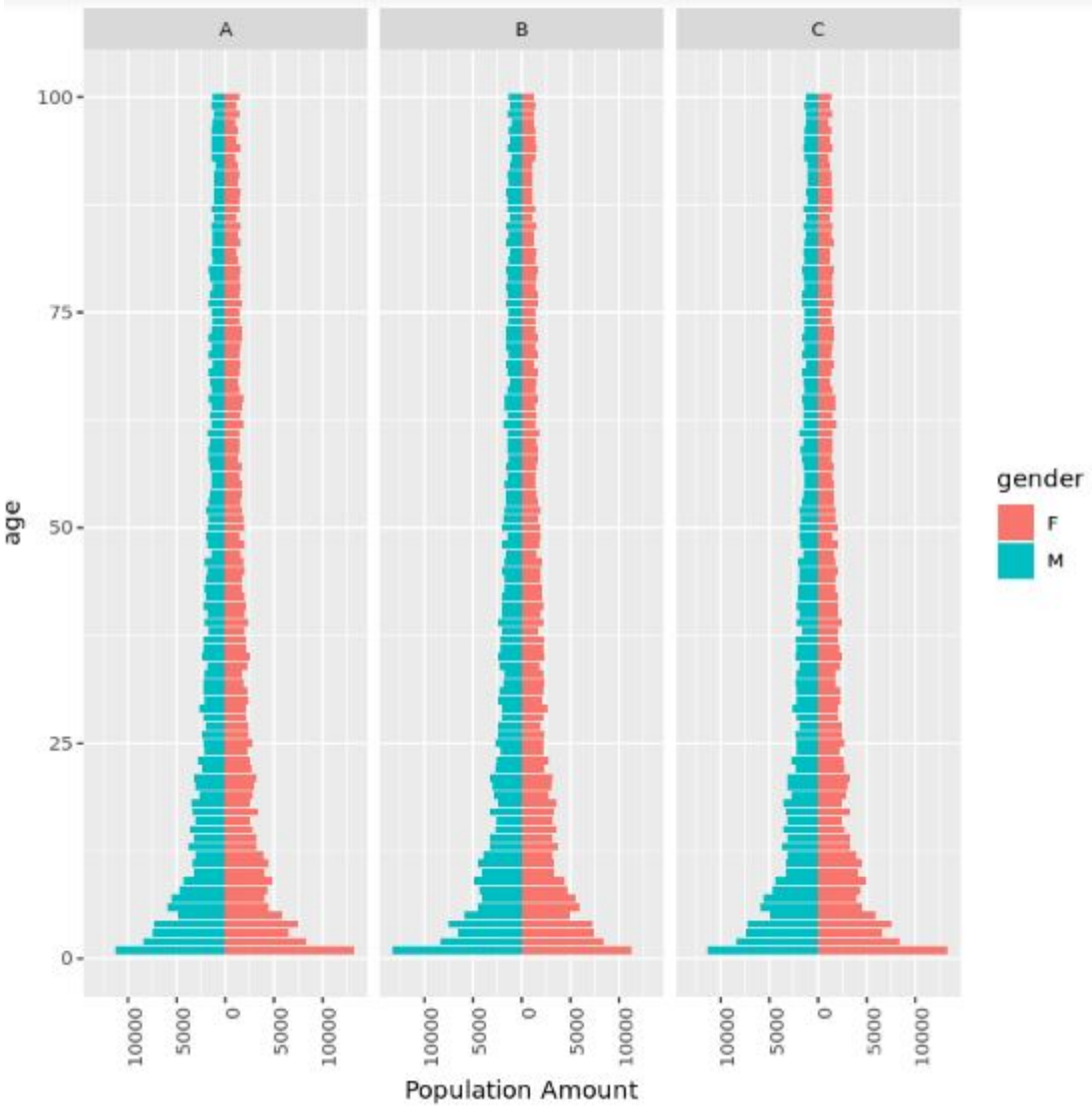

## Visualizing Multiple Populations using Faceting

One of the most powerful features of ggplot2 is its ability to create "small multiples," which are a series of smaller charts, each representing a subset of the data, allowing for easy visual comparison. When analyzing demographic data across several regions or countries, plotting multiple population pyramids side-by-side significantly enhances comparative analysis. This is achieved using the facet_wrap() function.

To demonstrate this, we first expand our original R data frame to include a new categorical variable named `country`, containing hypothetical countries A, B, and C. The data generation process is similar, ensuring each country has unique population counts across age and gender. Once the data is prepared, the visualization code remains largely the same, but we simply add the `facet_wrap(~ country)` layer.

The `facet_wrap()` function automatically splits the visualization into separate panels based on the values in the `country` variable, generating one population pyramid for each country. This technique is invaluable for identifying subtle differences in age structure, such as varying dependency ratios or generational gaps, across different geopolitical entities. We also include a minor theme adjustment (`theme(axis.text.x = element_text(angle = 90, hjust = 1))`) to handle potential overlap of x-axis labels in the smaller facets.

```
#make this example reproducible
set.seed(1)

#create data frame
data_multiple <- data.frame(age = rep(1:100, 6),
gender = rep(c("M", "F"), each = 300),
country = rep(c("A", "B", "C"), each = 100, times = 2))

#add population variable
data_multiple$population <- round(1/sqrt(data_multiple$age)*runif(200, 10000, 15000), 0)

#view first six rows of dataset
head(data_multiple)

# age gender country population
#1 1 M A 11328
#2 2 M A 8387
#3 3 M A 7427
#4 4 M A 7271
#5 5 M A 4923
#6 6 M A 5916

#create one population pyramid per country
ggplot(data_multiple, aes(x = age, fill = gender,
y = ifelse(test = gender == "M",
yes = -population, no = population))) +
geom_bar(stat = "identity") +
scale_y_continuous(labels = abs, limits = max(data_multiple$population) * c(-1,1)) +
labs(y = "Population Amount") +
coord_flip() +
facet_wrap(~ country) +
theme(axis.text.x = element_text(angle = 90, hjust = 1)) #rotate x-axis labels
```



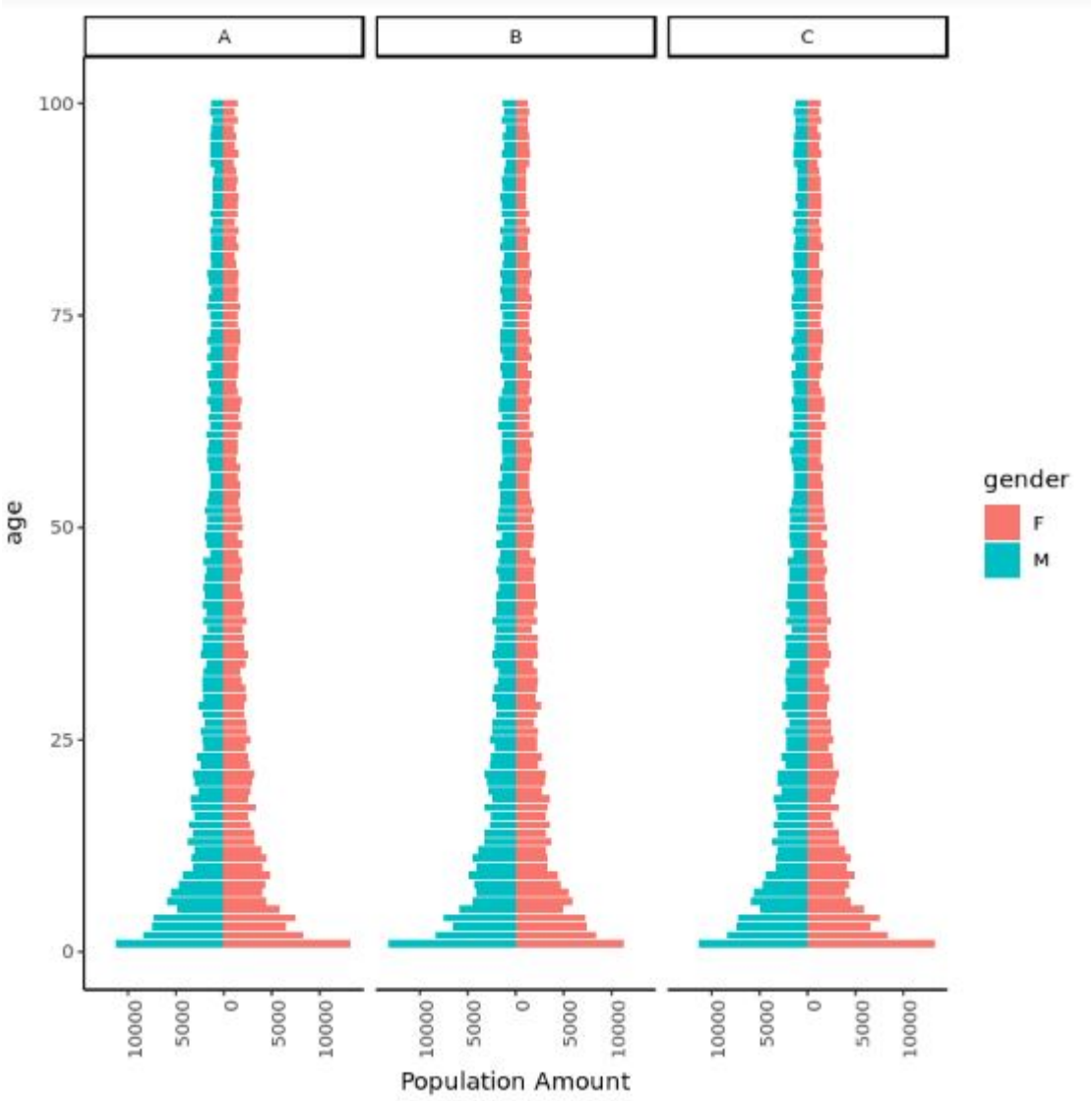

## Styling the Output: Applying Custom Themes

The final stage of creating a polished visualization is applying a suitable theme. ggplot2 offers several built-in themes that dramatically alter the non-data elements of the plot, such as background color, grid lines, and font styles. Choosing an appropriate theme ensures that the visualization is not only accurate but also professional and aesthetically pleasing, especially when integrating the chart into reports or publications.

For demographic visualizations, a minimalist style often works best, prioritizing the data bars over distracting background elements. The `theme_classic()` function is an excellent choice for achieving this. It removes the gray background and major grid lines, leaving only the primary axes and borders, which typically improves readability. This function is added as the penultimate layer to the R code block, applied globally to all facets created by facet_wrap().

Beyond built-in themes like theme_classic(), users can utilize external packages like `ggthemes`, which offer a much wider array of professional templates, including themes designed to mimic specific publications (e.g., `theme_economist` or `theme_wsj`). Modifying the theme is essential for transforming a statistical plot into a high-quality visualization suitable for formal presentation.

```
ggplot(data_multiple, aes(x = age, fill = gender,
y = ifelse(test = gender == "M",
yes = -population, no = population))) +
geom_bar(stat = "identity") +
scale_y_continuous(labels = abs, limits = max(data_multiple$population) * c(-1,1)) +
labs(y = "Population Amount") +
coord_flip() +
facet_wrap(~ country) +
theme_classic() +
theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




For a complete list of additional themes available to further customize your population pyramids, consult the official documentation for the `ggthemes` package.