PopHIVE Style Guide

2026-02-08

This style guide was created to ensure a consistent visual identity across PopHIVE products. Please see the instructions below to apply our pre-written functions to your data visualizations.

Setup

Load required packages

```
#load required packages
library(tidyverse)
library(ggplot2)
library(sf)
library(showtext)
library(scales)
library(maps)
library(ggstar)
```

Load all functions

All necessary functions are located in PopHIVE Insights repository: PopHIVE -> Insights -> PopHIVE_style_guide. Load these functions at the start of your session with the following code:

```
source("pophive_style_functions.R")
```

There are functions for 5 different types of graphs included:
* choropleth map * bubble map * line graph * dot plot * bar/ column chart

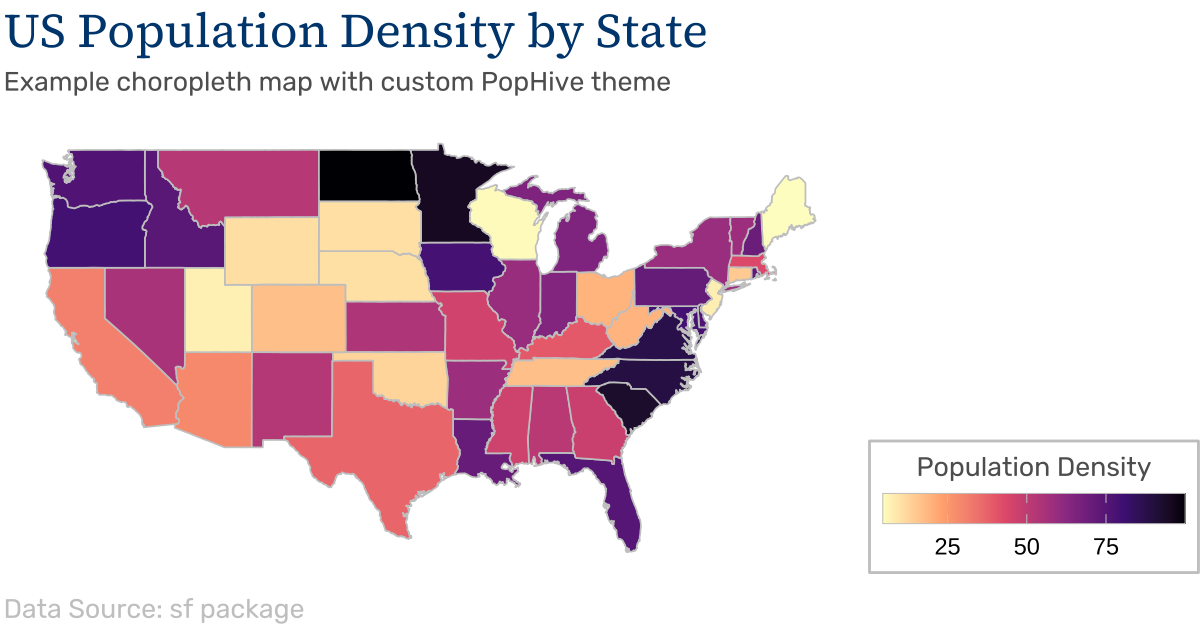
Examples

Choropleth Map

For the function to work correctly, your data must include a shapefile, which you can load with the "sf" package. Follow this format (the italicized text is what you will need to change):

ggplot(*data*) +
 geom_sf(aes(fill = *value_variable*), color = "gray", size = 0.3) +
  scale_fill_pophive_map(legend_title = "*Legend Title*") +
  theme_pophive_map() +
  labs(
    title = "*Title*",
    subtitle = "*Subtitle*",
    caption = "*Caption*")

```r
# Example

states <- st_as_sf(maps::map("state", plot = FALSE, fill = TRUE))
states$value <- runif(nrow(states), 0, 100)

ggplot(states) +
  geom_sf(aes(fill = value), color = "gray", size = 0.3) +
  scale_fill_pophive_map(legend_title = "Population Density") +
  theme_pophive_map() +
  labs(
    title = "US Population Density by State",
    subtitle = "Example choropleth map with custom PopHive theme",
    caption = "Data Source: sf package"
  )
```



Bubble Map

For the bubble map, you can use state level or county level data. You only need the state name and/or FIPS code.

There are several different options for the bubble map:

1. Coloring: There are separate helper functions for discrete and continuous color scales, depending on which is more appropriate for the data. If you want the bubbles to be a uniform color, you can use fill="#4e98e4" and omit the scale_color_x() functions.

2. Data resolution: There are helper functions to establish centroids for both state and county-level data.
3. Other aesthetics: Bubble size and transparency can be adjusted as described below.

Follow this format:
ggplot() +
 geom_sf(data = us_states, fill = "white", color = "gray", linewidth = 0.3) +
 geom_sf(data = *data*, aes(color = *value*, size = *size_var*), alpha = *0.6*, show.legend = TRUE) +

if you want to change the **color** aesthetic using a **continuous** scale:
 scale_color_pophive_bubble_continuous(legend_title = "*Legend Title*") +

if you want to change the **color** aesthetic using a **discrete** scale:
 scale_color_pophive_bubble_discrete(legend_title = "Category") +

if you want to change the **size** aesthetic:
 scale_size_pophive_bubble(legend_title = "*Legend Title*", range = c(2,15)) +

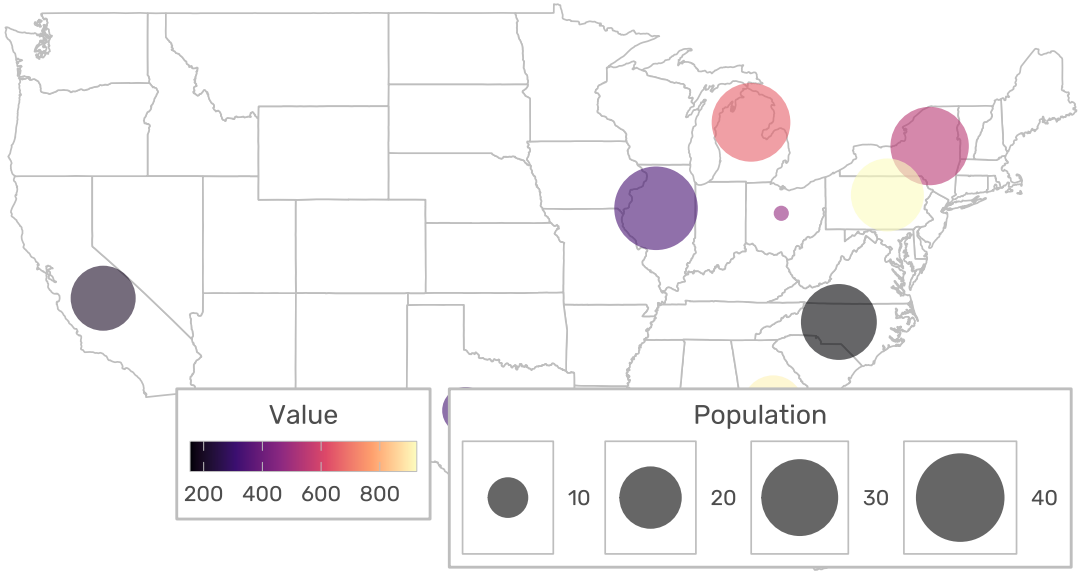
include for **all**:
 theme_pophive_bubble_map() +
 labs(
   title ="*Title*",
   subtitle = "*Subtitle*",
   caption = "*Caption*"
)

Note: currently debugging county-level bubble map functions. Will update accordingly.

```
# Bubble map with continuous scale and state-level data
# Get US state boundaries
us_states <- get_us_map_data(level = "state")

# Create sample data
bubble_data <- data.frame(
   location = c("California", "Texas", "Florida", "New York", "Illinois",
                "Pennsylvania", "Ohio", "Georgia", "Michigan", "North Carolina"),
   value = runif(10, 100, 1000),
   size_var = runif(10, 5, 50),
   category = sample(c("A", "B", "C"), 10, replace = TRUE)
 )

# Prepare bubble data with coordinates
bubble_sf <- prepare_bubble_data(bubble_data, location_col = "location", value_col = "value")

# Create bubble map with continuous color scale
ggplot() +
   geom_sf(data = us_states, fill = "white", color = "gray", linewidth = 0.3) +
   geom_sf(data = bubble_sf, aes(color = value, size = size_var),
           alpha = 0.6, show.legend = TRUE) +
   scale_color_pophive_bubble_continuous(legend_title = "Value") +
   scale_size_pophive_bubble(legend_title = "Population", range = c(2, 15)) +
   theme_pophive_bubble_map() +
   labs(
     title = "US Metropolitan Areas by Value",
     subtitle = "Bubble size represents population, color represents value metric",
     caption = "Source: Example data for demonstration purposes"
   )
```

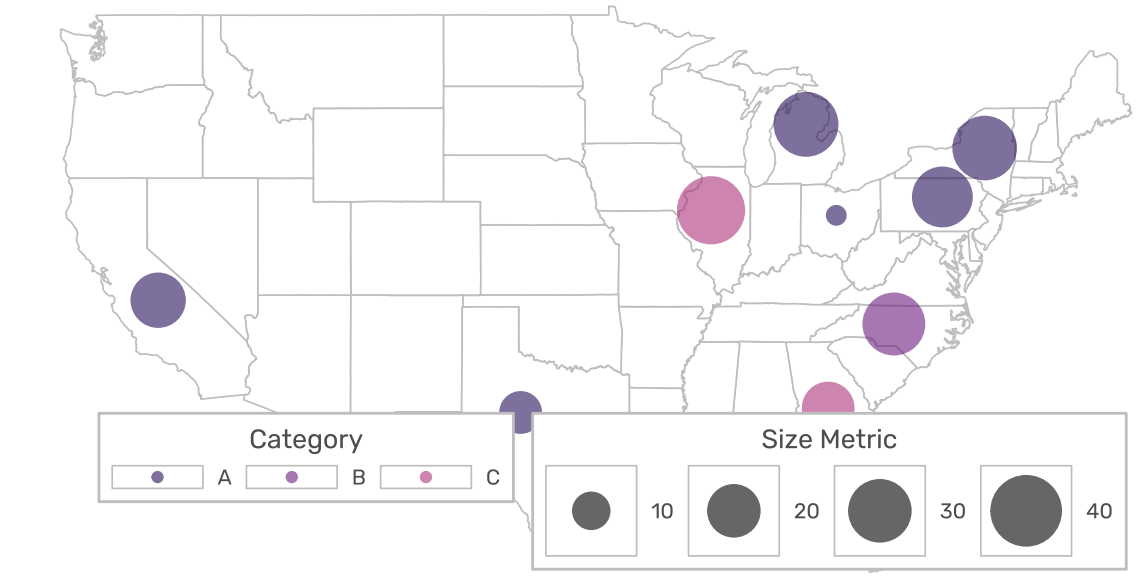
US Metropolitan Areas by Value

Bubble size represents population, color represents value metric



Source: Example data for demonstration purposes

```
# Bubble map with discrete scale and state-level data

ggplot() +
   geom_sf(data = us_states, fill = "white", color = "gray", linewidth = 0.3) +
   geom_sf(data = bubble_sf, aes(color = category, size = size_var),
           alpha = 0.6, show.legend = TRUE) +
   scale_color_pophive_bubble_discrete(legend_title = "Category") +
   scale_size_pophive_bubble(legend_title = "Size Metric", range = c(3, 12)) +
   theme_pophive_bubble_map() +
   labs(
     title = "US Cities by Category",
     subtitle = "Different categories shown in discrete colors",
     caption = "Source: Example categorical data"
   )
```



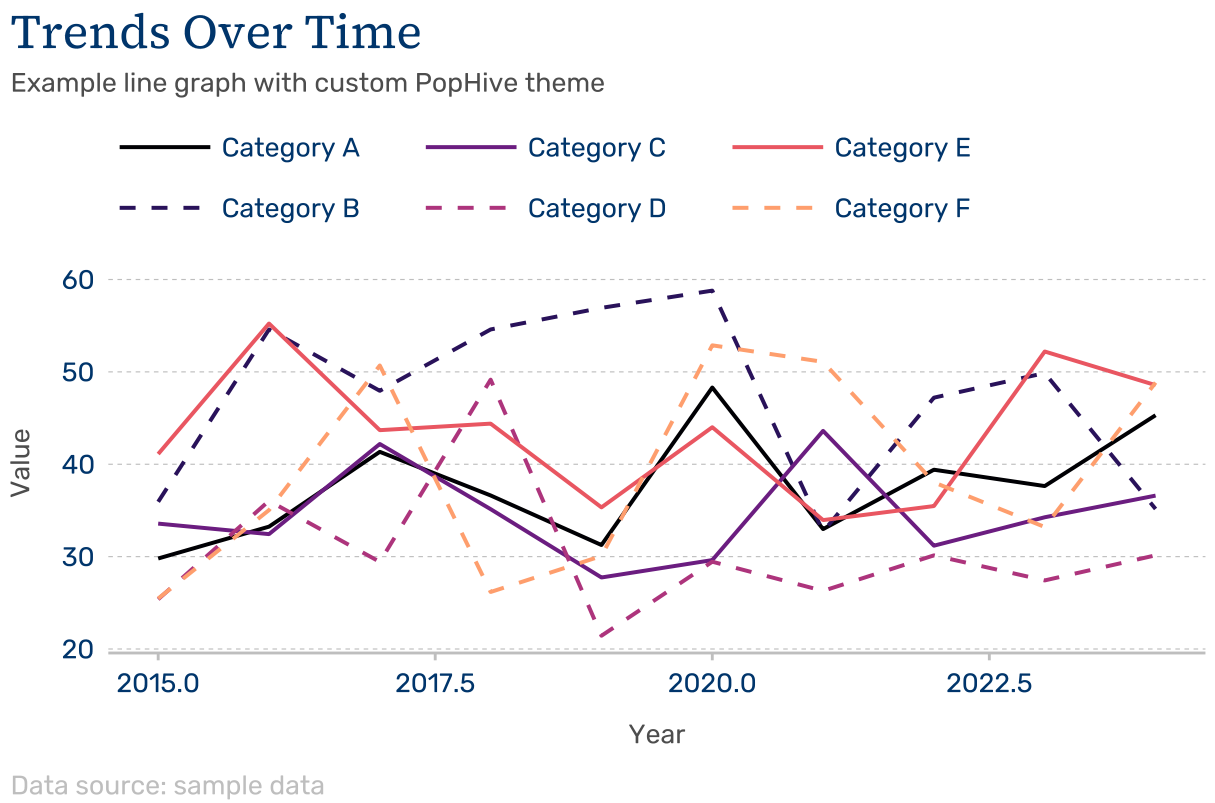
Line Graph

Note that for line graphs, you will need special helper functions for the color and linetype aesthetics. Follow this format:

ggplot(*date*, aes(x = *time_variable*, y = *value*, color = *category*, linetype = *category*)) +
geom_line(linewidth = 1) +
scale_x_date(date_breaks = "*frequency*", date_labels = *"%b %d, %Y"*) +
scale_y_continuous(labels = scales::*scale_type*) +
scale_color_pophive_line() +
scale_linetype_pophive_line() +
theme_pophive_line() +
labs(
title = "*Title*",
subtitle = "*Subtitle*",
x = "*X Axis Title*",
y = "*Y Axis Title*"
)

```
df_line <- data.frame(
  year = rep(2015:2024, 3),
  value = c(runif(10, 20, 50), runif(10, 30, 60), runif(10, 25, 55), runif(10, 20, 50), runif(10, 30, 6
  category = rep(c("Category A", "Category B", "Category C", "Category D", "Category E", "Category F"),
)
```

```r
ggplot(df_line, aes(x = year, y = value, color = category, linetype = category)) +
  geom_line(linewidth = 0.7) +
  scale_color_pophive_line() +
  scale_linetype_pophive_line() +
  theme_pophive_line() +
  labs(
    title = "Trends Over Time",
    subtitle = "Example line graph with custom PopHive theme",
    caption = "Data source: sample data",
    x = "Year",
    y = "Value"
  )
```



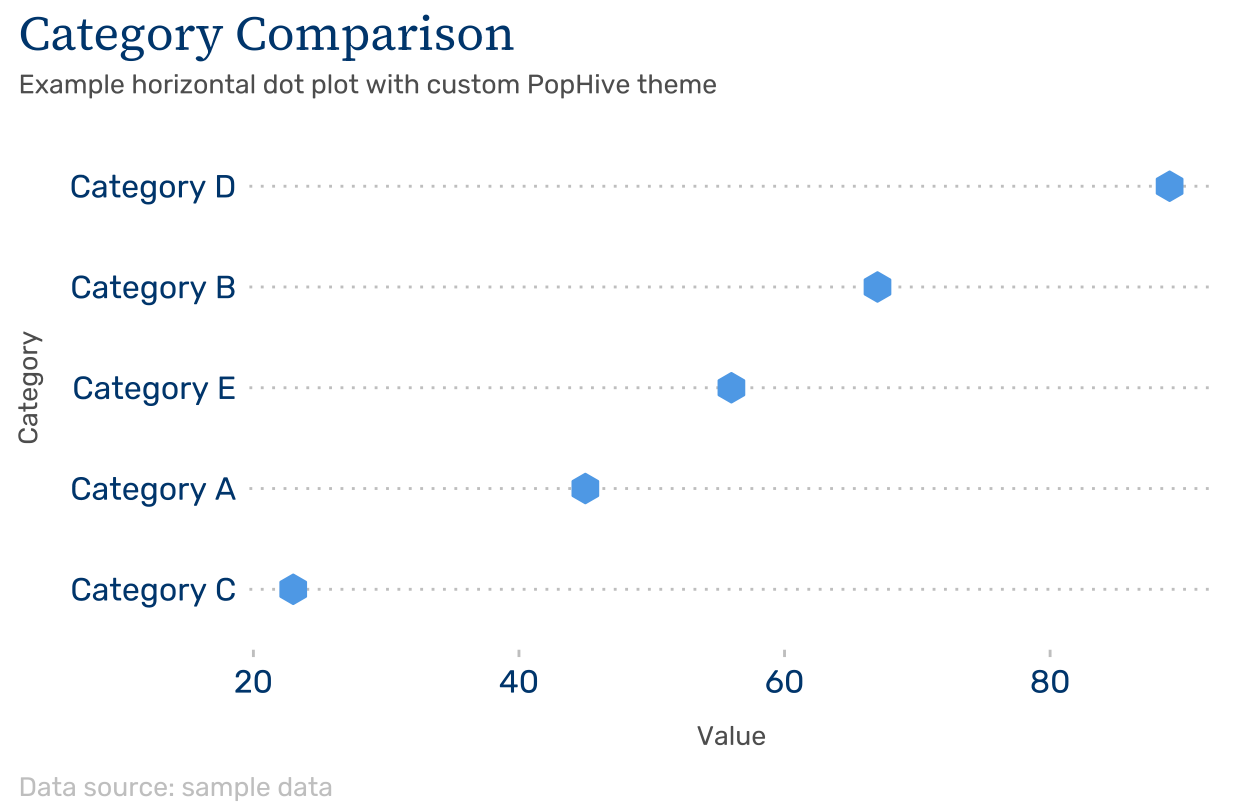
Dot Plot

For dot plots, follow this format:

ggplot(*data*, aes(x = *value*, y = *category*)) +
ggstar::geom_star(starshape = "hexagon", color = "#4e98e4", size = 4, fill = "#4e98e4") +
theme_pophive_dot() +
labs(
title = "*Title*",
subtitle = "*Subtitle*",
caption="*Caption*",
x = "*X Axis Title*",

y = “*Y Axis Title*”
)

```r
df_dot <- data.frame(
  category = c("Category A", "Category B", "Category C", "Category D", "Category E"),
  value = c(45, 67, 23, 89, 56)
)

ggplot(df_dot, aes(x = value, y = reorder(category, value))) +
  ggstar::geom_star(starshape = "hexagon", color = "#4e98e4", size = 4, fill = "#4e98e4") +
  theme_pophive_dot() +
  labs(
    title = "Category Comparison",
    subtitle = "Example horizontal dot plot with custom PopHive theme",
    caption="Data source: sample data",
    x = "Value",
    y = "Category"
  )
```



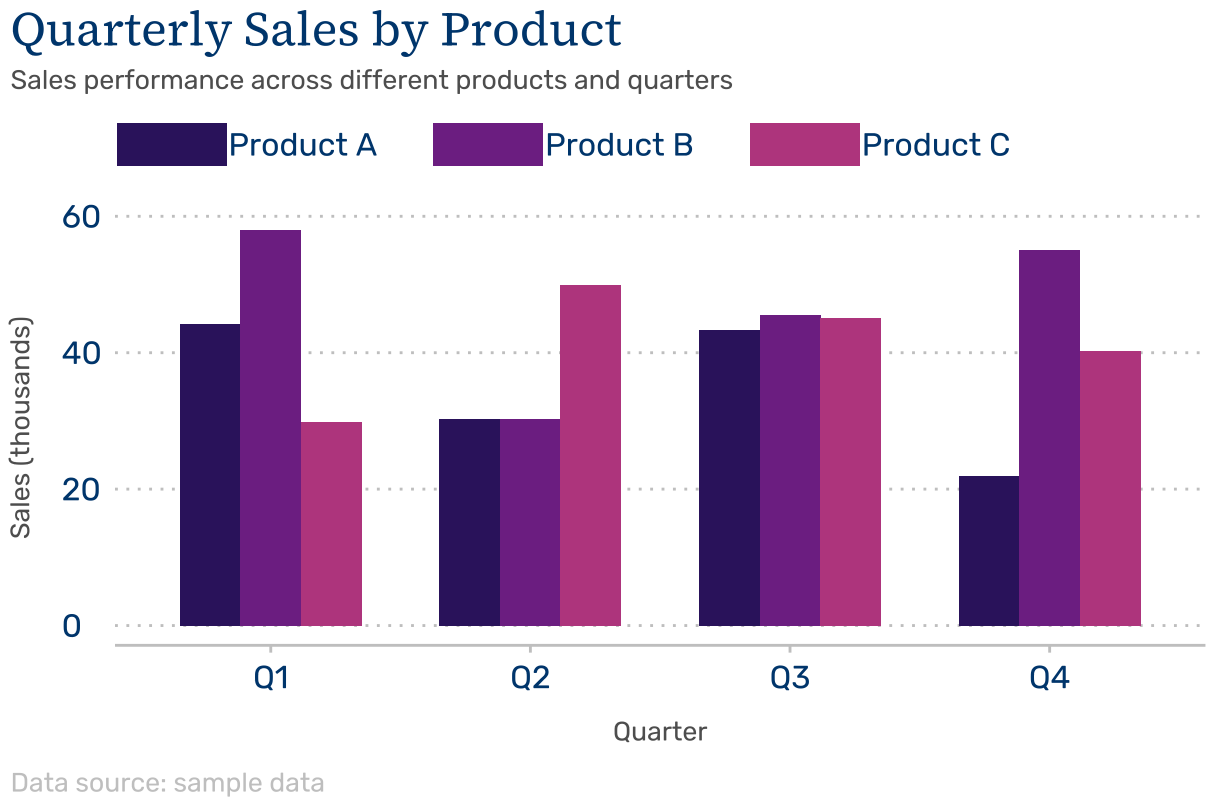
Bar/ Column Chart

For bar/ column charts, follow this format:

ggplot(*df_col*, aes(x = *category*, y = *value*, fill = *group*)) +
geom_col(position = “*dodge/stack/fill*”, width = 0.7) +

scale_fill_pophive_col() +
theme_pophive_col() +
labs(
  title = "*Title*",
  subtitle = "*Subtitle*",
  caption = "*Caption*",
  x = "*X Axis Title*",
  y = "*Y Axis Title*"
)

```
df_col <- data.frame(
  category = rep(c("Q1", "Q2", "Q3", "Q4"), 3),
  value = c(runif(4, 20, 50), runif(4, 30, 60), runif(4, 25, 55)),
  group = rep(c("Product A", "Product B", "Product C"), each = 4)
)

ggplot(df_col, aes(x = category, y = value, fill = group)) +
  geom_col(position = "dodge", width = 0.7) +
  scale_fill_pophive_col() +
  theme_pophive_col() +
  labs(
    title = "Quarterly Sales by Product",
    subtitle = "Sales performance across different products and quarters",
    caption = "Data source: sample data",
    x = "Quarter",
    y = "Sales (thousands)"
  )
```



Save your graphs

Use this format:

ggsave("*file name*.png", *plot name*, width = 12, height = 8, dpi = 300, bg = "white")