Laboratory Exercise

Data Visualization with ggplot2

EPID634 - Population Health Dashboard

Learning Objectives

By the end of this laboratory session, students will be able to:

- • Understand the grammar of graphics philosophy behind ggplot2
- • Create basic plots: scatter plots, line plots, bar charts, and histograms
- • Map variables to aesthetic properties (color, size, shape, transparency)
- • Use faceting to create multi-panel plots
- • Customize themes, labels, and titles for publication-quality graphics
- • Apply scales to control axis ranges, colors, and transformations
- • Create complex visualizations combining multiple geometries
- • Export plots in various formats for reports and presentations

Required R Packages

Install and load the following packages before beginning:

```
# Install packages

install.packages('tidyverse')  # Includes ggplot2

install.packages('scales')     # For better axis formatting

install.packages('patchwork')  # For combining plots

install.packages(lubridate)    # For dates and times

# Load libraries

library(tidyverse)   # For ggplot2 and data manipulation

library(scales)      # For formatting

library(patchwork)   # For plot composition

library(lubridate)   # For dates and times
```

Dataset Preparation

We'll create a comprehensive sales dataset for visualization practice:

```
# Create sample sales dataset

set.seed(123)

sales_data <- tibble(
  date = seq(as.Date('2024-01-01'), as.Date('2024-12-31'), by='day'),
  product = sample(c('Laptop', 'Phone', 'Tablet', 'Monitor', 'Keyboard'),
                   366, replace = TRUE),
  region = sample(c('North', 'South', 'East', 'West'),
                  366, replace = TRUE),
  sales = round(rnorm(366, mean = 50000, sd = 15000)),
  units = sample(10:100, 366, replace = TRUE),
  profit_margin = runif(366, 0.1, 0.4)
) %>%
  mutate(
    profit = sales * profit_margin,
    month = month(date, label = TRUE),
    quarter = quarter(date),
    day_of_week = wday(date, label = TRUE)
  )
```

PART 1: Basic Plot Types

1.1 Scatter Plots

Scatter plots show the relationship between two continuous variables.

Exercise 1.1: Creating Basic Scatter Plots

```
# Basic scatter plot
ggplot(sales_data, aes(x = sales, y = profit)) +
  geom_point()
# Add color by region
ggplot(sales_data, aes(x = sales, y = profit, color = region)) +
  geom_point()
# Customize point size and transparency
ggplot(sales_data, aes(x = sales, y = profit, color = region)) +
  geom_point(size = 3, alpha = 0.6)
```

Practice Task 1:

Create a scatter plot showing the relationship between units sold and profit margin, colored by product type.

1.2 Line Plots

Line plots are ideal for showing trends over time.

Exercise 1.2: Time Series Visualization

```
# Prepare data: aggregate by date
daily_sales <- sales_data %>%
  group_by(date) %>%
  summarise(total_sales = sum(sales))
# Basic line plot
ggplot(daily_sales, aes(x = date, y = total_sales)) +
  geom_line()
# Multiple lines by region
sales_by_region <- sales_data %>%
  group_by(date, region) %>%
  summarise(total_sales = sum(sales))
ggplot(sales_by_region, aes(x = date, y = total_sales, color = region)) +
  geom_line(size = 1)
```

Practice Task 2:

Create a line plot showing monthly total sales for each product category. Include both lines and points.

1.3 Bar Charts

Bar charts compare quantities across categories.

Exercise 1.3: Creating Bar Charts

```
# Prepare data: total sales by product
product_summary <- sales_data %>%
  group_by(product) %>%
  summarise(total_sales = sum(sales))
# Basic bar chart
ggplot(product_summary, aes(x = product, y = total_sales)) +
  geom_col()
# Colored bars with reordering
ggplot(product_summary, aes(x = reorder(product, total_sales),
                             y = total_sales, fill = product)) +
  geom_col() +
  coord_flip()  # Horizontal bars
# Grouped bar chart
product_region <- sales_data %>%
  group_by(product, region) %>%
  summarise(total_sales = sum(sales))
ggplot(product_region, aes(x = product, y = total_sales, fill = region)) +
  geom_col(position = 'dodge')
```

Practice Task 3:

Create a stacked bar chart showing total sales by region, with each bar divided by product category.

1.4 Histograms and Density Plots

Histograms show the distribution of a single continuous variable.

Exercise 1.4: Distribution Visualization

```
# Basic histogram
ggplot(sales_data, aes(x = sales)) +
  geom_histogram(bins = 30)
# Histogram with custom bins and color
ggplot(sales_data, aes(x = sales)) +
  geom_histogram(bins = 50, fill = 'steelblue',
                 color = 'black', alpha = 0.7)
# Density plot
ggplot(sales_data, aes(x = sales)) +
  geom_density(fill = 'lightblue', alpha = 0.5)
# Multiple distributions
ggplot(sales_data, aes(x = sales, fill = product)) +
  geom_density(alpha = 0.5)
```

Practice Task 4:

Create a histogram of profit margins, faceted by region (one panel per region).

PART 2: Aesthetic Mappings

2.1 Color, Size, and Shape

Aesthetics map data variables to visual properties.

Exercise 2.1: Working with Multiple Aesthetics

```
# Map multiple variables to aesthetics
ggplot(sales_data, aes(x = sales, y = profit,
                       color = product,
                       size = units,
                       shape = region)) +
  geom_point(alpha = 0.6)
# Fixed aesthetics (not mapped to data)
ggplot(sales_data, aes(x = sales, y = profit)) +
  geom_point(color = 'red', size = 3, alpha = 0.5)
```

Practice Task 5:

Create a scatter plot of sales vs units, where point color represents profit margin (use a gradient) and point size represents the day of the week.

PART 3: Faceting for Multi-Panel Plots

3.1 Facet Wrap and Facet Grid

Exercise 3.1: Creating Faceted Plots

```
# Facet by one variable
ggplot(sales_data, aes(x = sales, y = profit)) +
  geom_point() +
  facet_wrap(~ product)
# Facet grid (2 variables)
ggplot(sales_data, aes(x = sales, y = profit)) +
  geom_point(size = 1, alpha = 0.5) +
  facet_grid(region ~ product)
# Control facet arrangement
ggplot(sales_data, aes(x = sales)) +
  geom_histogram(bins = 30, fill = 'steelblue') +
  facet_wrap(~ product, ncol = 2, scales = 'free_y')
```

Practice Task 6:

Create a line plot showing daily sales trends, faceted by both quarter (rows) and region (columns).

PART 4: Themes and Customization

4.1 Built-in Themes

Exercise 4.1: Applying Themes

```
# Create a base plot
p <- ggplot(sales_data, aes(x = product, y = sales, fill = product)) +
  geom_boxplot()
# Try different themes
p + theme_minimal()
p + theme_classic()
```

```
p + theme_bw()
p + theme_dark()
```

4.2 Custom Labels and Titles

Exercise 4.2: Adding Labels

```
ggplot(sales_data, aes(x = sales, y = profit, color = product)) +
  geom_point(alpha = 0.6) +
  labs(
    title = 'Sales vs Profit Analysis',
    subtitle = 'Product Performance in 2024',
    x = 'Total Sales ($)',
    y = 'Profit ($)',
    color = 'Product Type',
    caption = 'Data source: Company Sales Database'
  )
```

Practice Task 7:

Create a well-labeled bar chart showing average profit margin by product, with appropriate title, subtitle, axis labels, and caption.

PART 5: Scales and Coordinates

5.1 Scale Functions

Exercise 5.1: Customizing Scales

```
# Color scales
ggplot(sales_data, aes(x = sales, y = profit, color = product)) +
  geom_point() +
  scale_color_brewer(palette = 'Set1')
# Manual color scale
ggplot(sales_data, aes(x = product, fill = product)) +
  geom_bar() +
  scale_fill_manual(values = c('Laptop' = '#E41A1C',
                               'Phone' = '#377EB8',
                               'Tablet' = '#4DAF4A',
                               'Monitor' = '#984EA3',
                               'Keyboard' = '#FF7F00'))
# Axis scales with formatting
library(scales)
ggplot(sales_data, aes(x = date, y = sales)) +
  geom_line() +
  scale_y_continuous(labels = dollar_format()) +
  scale_x_date(date_breaks = '1 month', date_labels = '%b')
```

Practice Task 8:

Create a scatter plot with a continuous color gradient representing profit margin, using a custom color palette (e.g., from red to green).

PART 6: Advanced Visualizations

6.1 Box Plots and Violin Plots

Exercise 6.1: Distribution Comparisons

```
# Box plot
ggplot(sales_data, aes(x = product, y = sales, fill = product)) +
  geom_boxplot()
# Violin plot
ggplot(sales_data, aes(x = product, y = sales, fill = product)) +
  geom_violin()
# Combination: violin + box + points
ggplot(sales_data, aes(x = product, y = sales)) +
  geom_violin(fill = 'lightblue') +
  geom_boxplot(width = 0.2, fill = 'white') +
  geom_jitter(alpha = 0.2, width = 0.1)
```

Practice Task 9:

Create a box plot showing profit distribution across different regions, with points overlaid and colored by product type.

6.2 Heatmaps

Exercise 6.2: Creating Heatmaps

```
# Prepare data for heatmap
heatmap_data <- sales_data %>%
  group_by(product, region) %>%
  summarise(avg_sales = mean(sales))
# Create heatmap
ggplot(heatmap_data, aes(x = region, y = product, fill = avg_sales)) +
  geom_tile() +
  scale_fill_gradient(low = 'white', high = 'darkblue') +
  geom_text(aes(label = round(avg_sales)), color = 'black')
```

Practice Task 10:

Create a heatmap showing average profit margin by product and day of the week.

Comprehensive Project: Sales Dashboard

Challenge: Create a comprehensive sales dashboard with multiple coordinated visualizations.

Requirements:

1. 1. Time series plot showing sales trends over time, with a smoothed trend line
2. 2. Bar chart comparing total sales by product, ordered from highest to lowest
3. 3. Faceted box plots showing sales distribution for each product across regions
4. 4. Scatter plot showing relationship between sales and profit, sized by units
5. 5. All plots should have consistent theme, appropriate labels, and professional formatting
6. 6. Combine all plots into a single dashboard layout using patchwork
7. 7. Export the final dashboard as a high-resolution PNG file

Starter Code:

```
library(patchwork)

# Plot 1: Time series

p1 <- # Your code here

# Plot 2: Bar chart

p2 <- # Your code here

# Plot 3: Faceted box plots

p3 <- # Your code here

# Plot 4: Scatter plot

p4 <- # Your code here

# Combine plots

dashboard <- (p1 + p2) / (p3 + p4) +

  plot_annotation(

    title = 'Sales Performance Dashboard 2024',

    theme = theme(plot.title = element_text(size = 18, face = 'bold'))

  )

# Save

ggsave('sales_dashboard.png', dashboard,

       width = 16, height = 10, dpi = 300)
```

Best Practices for Data Visualization

Design Principles

- Choose the right chart type for your data (continuous vs categorical)
- Use color purposefully - not just for decoration
- Keep it simple - avoid chart junk and unnecessary elements
- Make text readable - use appropriate font sizes
- Order categories meaningfully (alphabetically or by value)
- Start y-axis at zero for bar charts (unless justified)
- Use consistent colors across related visualizations

Technical Tips

- Build plots incrementally - test each layer
- Save plot objects to variables for reuse and modification
- Use faceting instead of overplotting when possible
- Set themes globally with theme_set() for consistency
- Export at high resolution (300 dpi) for publications
- Use appropriate aspect ratios (width to height)

Useful Resources

- ggplot2 documentation: https://ggplot2.tidyverse.org/
- R Graphics Cookbook: https://r-graphics.org/
- Data Visualization cheat sheet: https://github.com/rstudio/cheatsheets
- Color palettes: ColorBrewer, viridis, wesanderson packages