Anscombe's Quartet: Visualization & Regression

EA30 Reflection

October 8, 2025

This handout explores Anscombe's Quartet, a famous dataset demonstrating why data visualization is essential before fitting statistical models. Through guided exercises, students will discover how four datasets with identical summary statistics can have dramatically different patterns.

Introduction: Why Visualization Matters

Anscombe's Quartet is a famous dataset created by statistician Francis Anscombe in 1973 to demonstrate the importance of visualizing data before analyzing it. Four datasets have nearly identical summary statistics (mean, variance, correlation, regression line) but look completely different when graphed!

Key Question to Consider: Can we trust summary statistics alone?

Learning Goal: Understand why data visualization is essential before fitting statistical models.

Setup: Loading Required Packages

We need several R packages for data manipulation and visualization: `ggplot2` for creating visualizations, `dplyr` for data manipulation, and `tidyr` for reshaping data.

Help Topics:
`?library`
`?install.packages`

Data Preparation

The Anscombe dataset comes in "wide" format with separate columns for each set (x1-y1, x2-y2, etc.). We'll transform it to "long" format for easier analysis. We can create 4 dataframes, each with different predictor and response variables, then we can combine them into a single dataframe, which I called df, using rbind().

```
# Creating a dataframe with one set
  d1 <- data.frame(x = anscombe$x1, y = anscombe$y1, set = "1")
```

Pseudocode:

1. Load anscombe dataset
2. Extract each x-y pair
3. Add set identifiers
4. Combine with `rbind()`

Help Topics:
`?anscombe`
`?data.frame`
`?rbind`

Optional: Using tidyr for Data Reshaping

The method above works well, but there's a more elegant way using `tidyr`! The `pivot_longer()` function can transform wide data to long format in fewer steps.

Concept: Instead of manually creating four separate data frames, we can:

Advanced Technique: The `tidyr` package provides powerful functions for reshaping data.

Help Topics:
`?pivot_longer`
`?separate`
`?mutate`

1. Add a row ID to track observations
2. Pivot all x columns into one column and all y columns into another
3. Separate the set number from the variable name
4. Clean up the result

Here's what the tidyr approach looks like:

```
# Alternative approach using tidyr
df_tidy <- anscombe %>%
  mutate(obs = row_number()) %>%
  pivot_longer(
    cols = -obs,
    names_to = c(".value", "set"),
    names_pattern = "(.)(.)"
  ) %>%
  select(-obs)
```

Exercise 1b (Optional): Comparing Approaches

For advanced students, try implementing the tidyr approach:

1. Type `View(anscombe)` to see the original wide format
2. Implement the tidyr code above
3. Use `identical(df, df_tidy)` to verify both methods produce the same result
4. Which approach do you find more intuitive? Why?

Key Insight: `pivot_longer()` with `names_pattern` recognizes that "x1" means "x variable, set 1" and automatically separates them. The `.value` placeholder tells it to create separate columns for x and y.

Exercise 1: Exploring the Data Structure

Before moving on, answer these questions:

1. How many total observations are in the combined `df` dataset? (Hint: use `nrow(df)`)
2. What are the column names? (Hint: use `names(df)`)
3. Look at the first few rows of dataset 1. What do you notice? (Hint: use `head(subset(df, set == "1"))`)

Bonus: If you completed Exercise 1b above using tidyr, compare the structure of both datasets using `str(df)` and `str(df_tidy)`.

Fitting Linear Models

We fit a separate linear regression model (y ~ x) to each of the four datasets and extract key statistics: slope, intercept, and RÂš.

Pseudocode:

```
1. Filter data for each set
2. Fit model with lm()
3. Extract coefficients
4. Extract $R^2$ values
5. Create summary table
```

Help Topics:
`?lm`
`?coef`
`?summary.lm`
`?subset`

Exercise 2: Understanding Model Output

PRACTICE INTERPRETING regression output:

1. Use `summary(fit1)` to see the detailed output for dataset 1. What is the p-value for the slope coefficient?
2. What does the slope tell us about the relationship between x and y?
3. Try running `plot(fit1)` – what diagnostic plots appear?

Basic Visualization

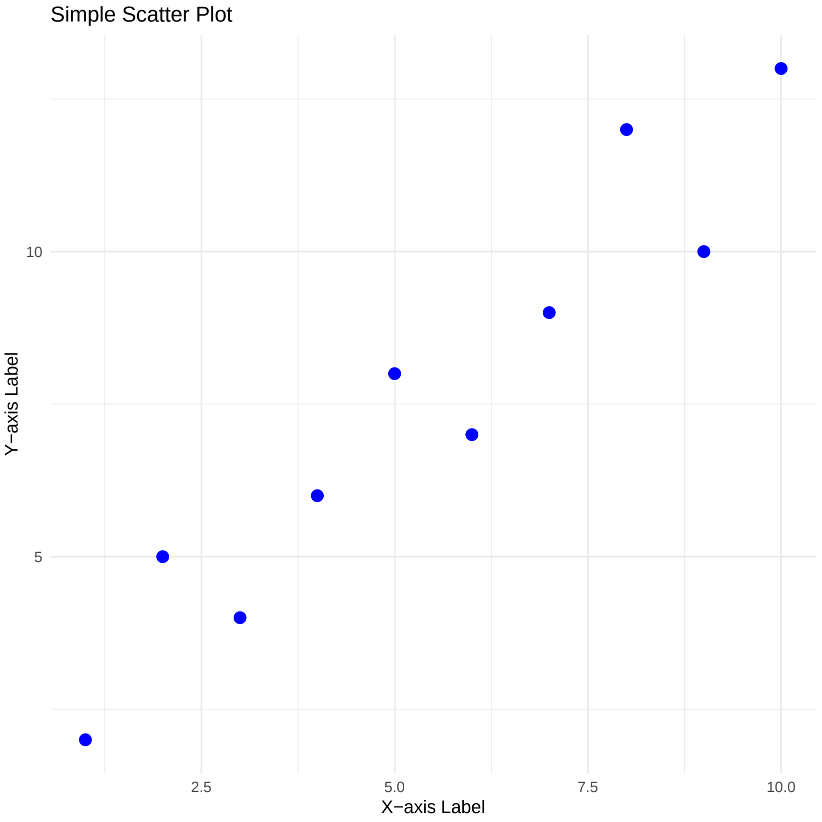
Help Topics:
`?ggplot`
`?geom_point`
`?geom_smooth`
`?facet_wrap`

WHAT IS `ggplot2` – it's a way to visualize data in R that provides a great deal of flexibility.

```
# Load ggplot2 package
library(ggplot2)

# Create a small example dataset
demo <- data.frame(
  x = 1:10,                          # X-axis values
  y = c(2, 5, 4, 6, 8, 7, 9, 12, 10, 13)  # Y-axis values
)

# Create a basic scatter plot
ggplot(demo, aes(x = x, y = y)) +   # Map x and y variables
  geom_point(                       # Add points to the plot
    color = "blue",                 # Point color
    size = 3                         # Point size
  ) +
  labs(                              # Add labels
    title = "Simple Scatter Plot", # Plot title
    x = "X-axis Label",            # X-axis label
    y = "Y-axis Label"             # Y-axis label
  ) +
  theme_minimal()                  # Apply a minimal theme for clean look
```



Simple Scatter Plot Example

Formatting Options Explained

- **aes(x = x, y = y)**: maps dataset columns to the plot axes.
- **geom_point()**: adds points; can set `color`, `size`, `shape`, etc.
- **labs()**: sets labels like title, subtitle, x-axis, y-axis, and caption.
- **theme_minimal()**: changes plot appearance. Alternatives include `theme_bw()`, `theme_classic()`, etc.
- **Layering with +**: ggplot2 uses + to combine layers like points, lines, labels, and themes.

Notes for EA30

- Each element (`geom_point`, `labs`, `theme_minimal`) can be customized independently.
- You can experiment with colors, sizes, shapes, and themes to see how the plot changes.
- ggplot2 encourages a "grammar of graphics" approach: map data first (`aes`), then add layers and formatting.

Now we create a 2 x 2 grid of scatterplots with regression lines to visualize all four datasets. The plot uses `facet_wrap()` to create separate panels for each dataset.

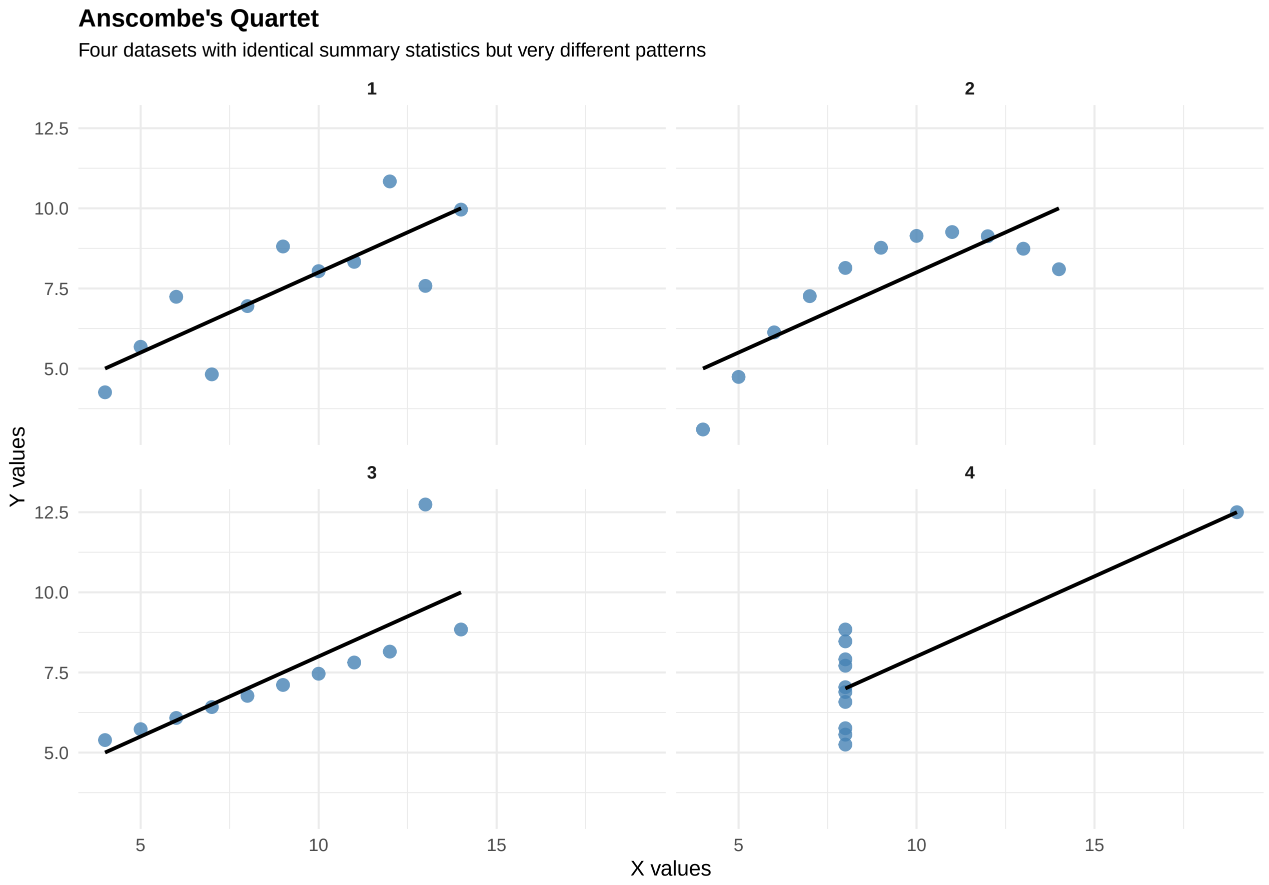


Figure 1: Scatterplots of all four Anscombe datasets with fitted regression lines. Notice the dramatic differences in patterns despite identical statistical summaries.

Exercise 3: Visual Interpretation

Look carefully at the four plots and answer:

1. Which dataset appears to have a truly linear relationship?
2. Which dataset has a curved (non-linear) pattern?
3. Which datasets have outliers? How many in each?
4. Based on visualizations alone, which datasets would be appropriate for linear regression? Why?

Summary Statistics

Notice: All four datasets have nearly identical statistics! Slope $\sim$ 0.50, Intercept$\sim$ 3.00, R^2 $\sim$ 0.67

Table 1: Regression statistics for all four datasets

set	slope	intercept	r2
1	0.5	3.000	0.667
2	0.5	3.001	0.666
3	0.5	3.002	0.666
4	0.5	3.002	0.667

Exercise 4: Comparing Statistics

Calculate additional summary statistics:

1. Calculate the mean of x for each dataset. Are they the same? (Hint: `tapply(df$x, df$set, mean)`)
2. Calculate the mean of y for each dataset using the same approach
3. Calculate the correlation for each dataset

Question: If the means, correlations, and regression lines are all nearly identical, why do the datasets look so different?

Annotated Visualization

Exercise 5: Customizing Visualization

Critical Observations:
Set 1: Linear (appropriate)
Set 2: Non-linear (curved)
Set 3: Linear + outlier
Set 4: Outlier creates false correlation

Practice modifying the plot:

1. Change the point color from "steelblue" to another color
2. Modify the point size – try making them bigger or smaller
3. Change the title to something creative
4. Try `theme_bw()` or `theme_classic()` instead

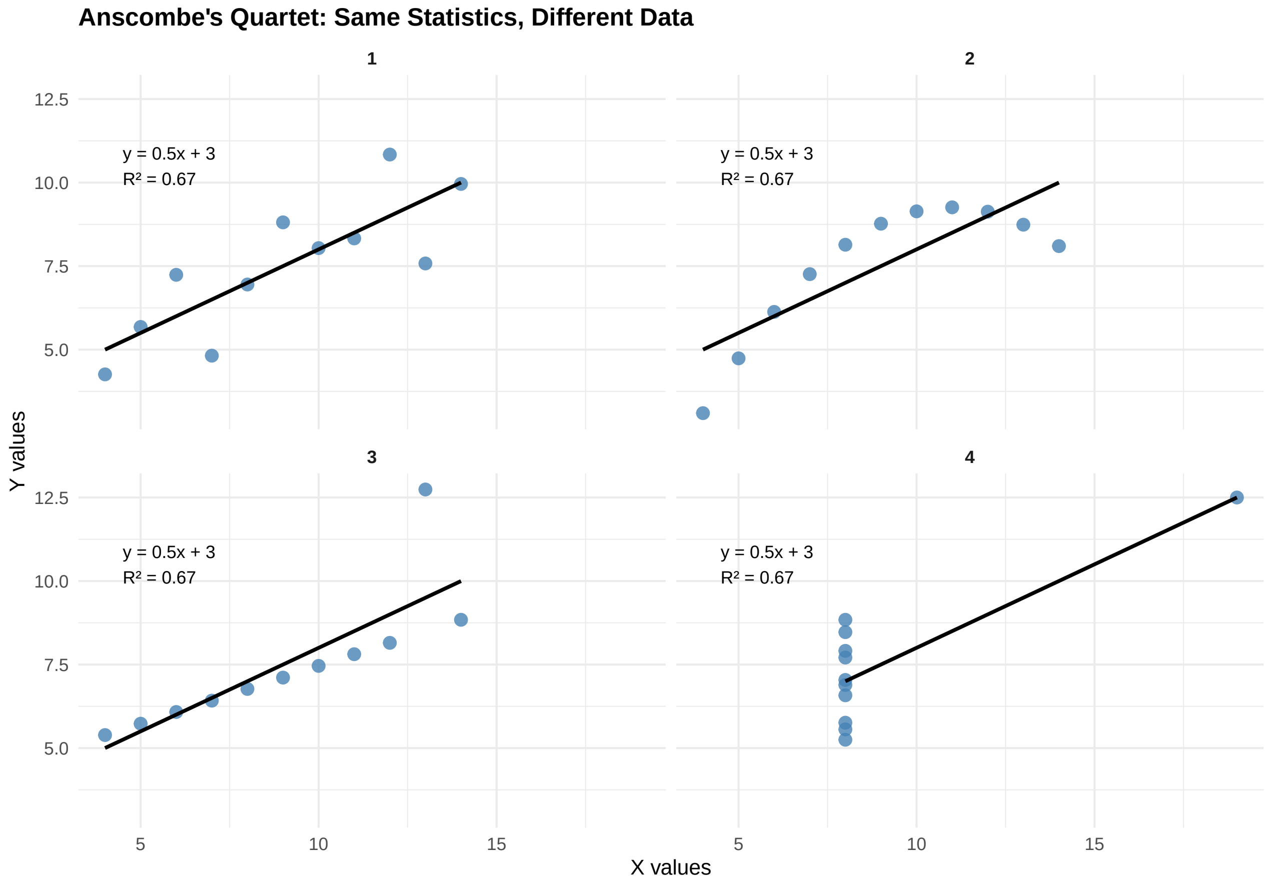


Figure 2: Complete visualization with regression equations displayed on each panel. Despite identical equations, the datasets reveal very different underlying patterns.

Residual Analysis

Residuals help us diagnose whether our linear model is appropriate. Good models have randomly scattered residuals around zero with no clear patterns.

Key Concept: Residuals are the differences between observed and predicted y values. Random scatter = good model fit!

Exercise 6: Interpreting Residual Plots

Study the residual plots and answer:

1. Which dataset has residuals that appear randomly scattered? (Good!)
2. Which dataset shows a clear pattern in the residuals? (Linear model inappropriate)
3. Which datasets have extreme residuals (very far from zero)?
4. What do these plots reveal that R^2 did not?

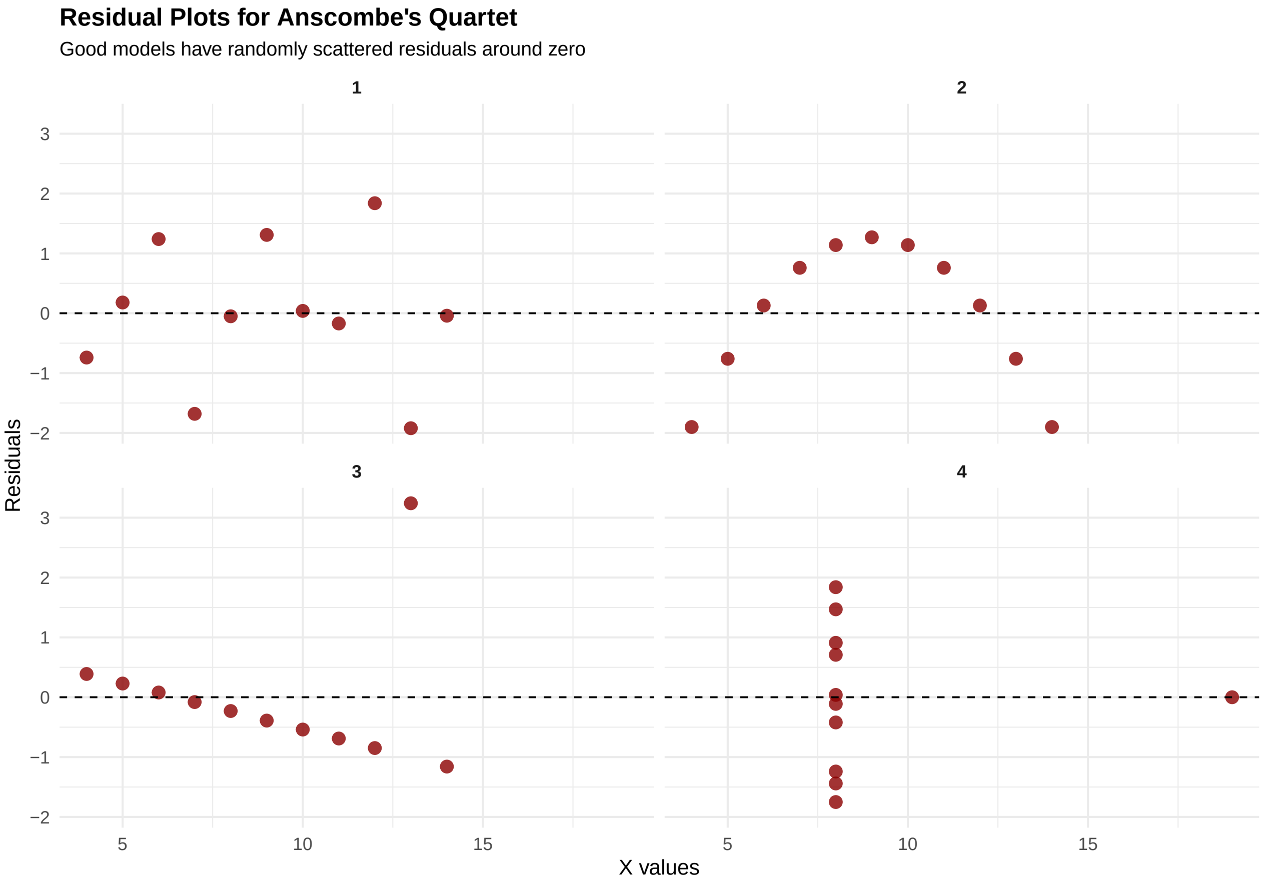


Figure 3: Residual plots reveal problems invisible in summary statistics. Set 1 shows random scatter (good), Set 2 shows clear curvature (bad), and Sets 3 & 4 show extreme outliers.

Challenge: Impact of Outliers

Exercise 7: Removing Outliers

For Dataset 3 (which has one outlier):

1. Create a new dataset excluding the outlier point (x=13)
2. Fit a new regression model to this filtered data
3. Compare the new slope and R^2 to the original
4. Create a scatterplot with the new regression line

Starter hints:

```
df3_no_outlier <- subset(df, set == "3" & x != ?)
fit3_no_outlier <- lm(y ~ x, data = df3_no_outlier)
```

Question: How much did removing one point change your results? What does this tell you about the influence of outliers?

Final Reflection

Synthesize what you've learned:

1. **Main Lesson:** In your own words, what is the key message of Anscombe's Quartet?

2. **Real-World Application:** Think of a situation where someone might make a mistake by looking only at statistics without visualizing data.

3. **Best Practices:** What steps should you ALWAYS take before trusting a linear regression model?

4. **Going Forward:** What other diagnostic tools might you use when analyzing real data?

Key Takeaways

- Always visualize your data before analysis
- Summary statistics can hide important patterns
- Outliers can dramatically affect regression results
- Check model assumptions, don't just trust RÂš values
- Residual plots help diagnose model problems
- Different data patterns require different analytical approaches

Additional Resources

- **R Documentation:** Use `?function_name` in console
- **ggplot2 Cheat Sheet:** `https://rstudio.github.io/cheatsheets/`
- **Datasaurus Dozen:** A modern extension with even more diverse patterns