# Linear Regression Analysis

Christopher Weber, PhD

2026-03-29

# Table of contents

# Introduction

This is the course reader for POL 682. The reader provides a comprehensive overview of linear regression techniques, including simple and multiple linear regression, with practical examples and visualizations.

## Overview

Linear regression is a fundamental statistical technique used to model the relationship between a dependent variable and one or more independent variables. This book covers:

- Basic concepts of linear regression
- Simple linear regression
- Deriving the OLS Estimator
- Gauss-Markov Assumptions and Theorem
- Multiple linear regression
- Model diagnostics and validation
- Numerous applications with real data

## Prerequisites

To follow along with the examples in this book, you should have:

- Basic knowledge of statistics
- Completed POL 681 or equivalent
- Basic understanding of R programming
- Understanding of mathematical notation
- Familiarity with matrix algebra is helpful but not required (there is also a chapter in this reader)

## Structure

The book is organized into several chapters:

1. **Simple Linear Regression**: Introduction to modeling relationships between two variables
2. **Ordinary Least Squares**: Deriving the OLS estimator
3. **Model Diagnostics**: Checking assumptions and validating models
4. **Applications and Case Studies**: Practical examples with real data

# Part I

# Part I: Foundations

# Chapter 1

# Understanding Linear Regression

## 1.1 Overview

Linear regression is one of the most widely used statistical methods for several reasons. First, in the broader scheme of statistical techniques, linear regression is relatively straightforward – as well as foundational – to understand more complex methods. It's an ideal starting point to learn more advanced techniques.

Second, linear regression is used in a variety of contexts. It's invaluable for descriptive analysis – describing the relationship between variables in a sample. It's useful for exploratory data analysis – identifying potential realtionships or patterns in the data. It's essential for inference as well – describing the relationship between variables in a population, based on a sample. It is routinely used in hypothesis testing and predictive modeling.

Linear regression is applicable in many contexts, helping us distill complex relationships between "observed" and "experimental" data. The technique is used to estimate the (linear) relationships between two or more variables. At its core, linear regression follows this form.

The linear model follows this basic functional form.

$y_i = \alpha + \beta x_i$

$\alpha$ represents the intercept. The point at which the regression line crosses the $y$ axis, or equivalently, the value of $y$ when $x = 0$.

$\beta$ represents the slope, the change in $y$ for every unit change in $x$. And more formally,

$\partial y / \partial x = \beta$

Because the relationship between $x$ and $y$ is imperfect – **stochastic** rather than **deterministic** – we add an error term, $\epsilon_i$.

$$y_i = \alpha + \beta x_i + \epsilon_i$$

$\epsilon_i$ represents the errror term. Often, we'll often assume that $\epsilon_i \sim N(0, \sigma^2)$, normally distributed with mean zero and constant variance $\sigma^2$.

In this example, $y$ is an outcome, or **endogeneous** variable, whereas $x_i$ is a predictor, or **exogeneous** variable. If we have multiple predictors, we could write the equation as:

$y_i$=$\alpha$+$\beta_1 x_{1i}$ + $\beta_2 x_{2i}$ + … + $\beta_k x_{ki}$ +$\epsilon_i$

Or more compactly

$y_i$=$\alpha$+$\sum_{j=1}^{k} \beta_j x_{ji}$ +$\epsilon_i$

In these examples, we are modeling the relationship between y and x. We are modeling the *probability* of y given x.

$p(y|x_1, x_2, ..., x_k)$, the conditional distribution of $y$ given $x_1, x_2, ..., x_k$.

$$p(y|x_1, x_2, ..., x_k) = f(x_1, x_2, ..., x_k; \alpha, \beta_1, \beta_2, ..., \beta_k)$$

And the key question to address is **what is the expected value of $y$ given $x_1, x_2, ..., x_k$?**. In repeated samples, what is the expected value of $y$ given fixed values $x_1, x_2, ..., x_k$?

$$E(y|x_1, x_2, ..., x_k) = \alpha + \sum_{j=1}^{k} \beta_j x_j$$

### 1.1.1 Important Considerations

In this class, we'll derive the OLS estimator, one approach to estimate the parameters *alpha* and $b_k$. The key to linear regression is to typically just **write out** the model. Yet writing and estimating a linear equation does not mean the model is correct or accurate. The reasons are multifaceted. First, **confounding** may be a concern. Let's say $x$ may be related to $y$, but there is a common variable , $u$, that affects scores on both $x$ and $y$. The linear regression – as just a statistical procedure – can be defined to estimate the **wrong** relationships. In this case, $x \not\perp y|u$.

Second, there is the concern of ** exogeneity and causation.** Linear regression is a mathematical procedure. It doesn't inherently allow one to make a causal statement. The slopes are intimately related to the correlation coefficient. For the same reason that correlated variables do not imply a causal relationship, the slope parameters in a linear regression also do not imply causality.

Third, there the issue off **indirect effects** and mediation, a *process based argument.* $x$ may be related to $y$, due to the intervening mechanisms $z$, so $x \rightarrow z \rightarrow y$. A relationship may be indirect. It is difficult to sort these out using a regression model. In most applications, we observe the outcome of a process. It is often difficult to make claims about the process itself.

**Linear regression is a technique to estimate the relationships between multiple variables. A slope represents a change in y for a unit change in x. The quality of these estimates depend on multiple things, including accurate measurement, correct model specification, and no confounding. Linear regression, like many statistical approaches, relies on several strong assumptions, which we'll examine in this class.**

## 1.2 `lm`

### 1.2.1 An Example with Spotify Data

We'll use a number of data sets in this class. Let's start out with a public one, *Spotify* music data. These public data include song popularity in 2025. The data include `track`, `artist`, and `album` level information. The data are freely available for download on `kaggle.com` and are saved in the class Github repository.

In order to load a package in `R`, it is necessary to install that package. Most packages are available on `CRAN` or `Github`.

```
install.packages("tidyverse")
```

After installation, the package can be loaded into memory in subsequent `R` sessions.

```
load(tidyverse)
```

**`tidyverse` is a collection of packages that all follow a similar structure. If we use `library(tidyverse)` all these packages are available in the R session. You can find a number of useful cheatsheets here, https://posit.co/resources/cheatsheets/**

**Typically, it's best to only load those packages that are being used, as loading too many packages can lead to conflicts. Like, if two packages use the same name for a function, R will only evaluate the one most recently loaded. This can cause code to break, and a lot of frustration, so it's best to only load those packages that are needed..**

```
library(tidyverse)
library(dplyr)
library(tidyr)
library(ggplot2)
download.file("https://raw.githubusercontent.com/crweber9874/linear_regression_class/main/data/sp
```

```
              destfile = "spotify_data.csv")
spotify_data <- read.csv("spotify_data.csv", row.names = NULL)
cat("The variable names (column names) are:\n")
```

```
The variable names (column names) are:
```

```
names(spotify_data)
```

```
 [1] "track_id"           "track_name"         "track_number"
 [4] "track_popularity"   "explicit"           "artist_name"
 [7] "artist_popularity"  "artist_followers"   "artist_genres"
[10] "album_id"           "album_name"         "album_release_date"
[13] "album_total_tracks" "album_type"         "track_duration_min"
```

### 1.2.2 Data Exploration and Research Questions

There's a lot in these data. Let's just explore a simple relationship – what predicts the popularity of a music track? There are a few variables to examine

- `track_popularity`: Popularity of the track (0-100)
- `explicit`: TRUE/FALSE indicating whether the track has explicit content
- `track_number`: Track number on the album
- `artist_followers`: Number of followers of the artist

**Research Questions**

1) Is there a relationship between the number of followers an artist has and the popularity of their tracks?
2) Does the track number on an album influence its popularity? Do popular tracks appear earlier in an album
3) Does explicit content affect track popularity?

**`ggplot` is commonly used for graphics in `R`. ggplot adopts a syntax in which aesthetic characteristics are updated and modified through the `+` operator. The first layer defines the data and relevant variables. The subsequent `geom()` layers describe what to plot. The remaining layers style the axes, change fonts, and so forth. See https://ggplot2.tidyverse.org/**

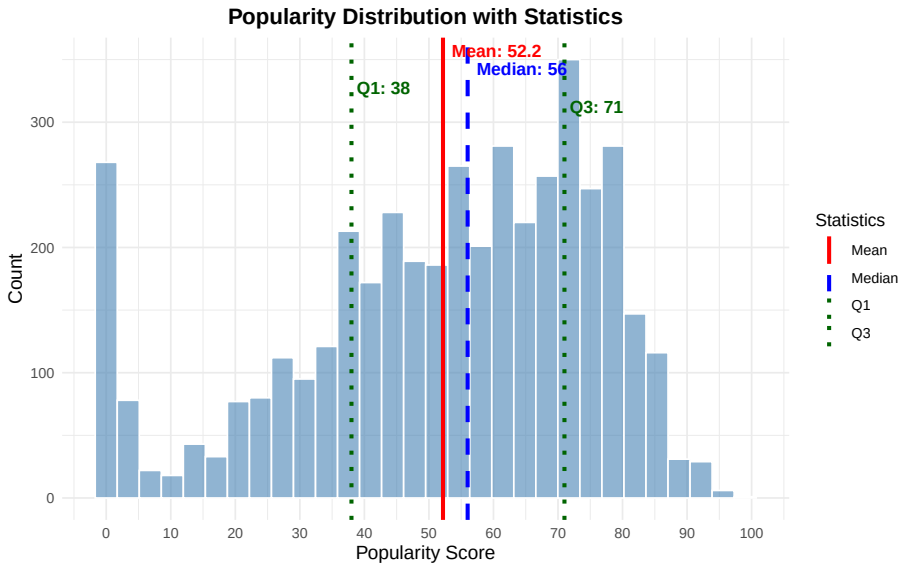
```
library(ggplot2)
library(dplyr)

dat <-
  suppressWarnings(spotify_data |>
  mutate(track_popularity = as.numeric(track_popularity),
         artist_followers = as.numeric(artist_followers),
         explicit = ifelse(explicit == TRUE, 1, 0),
         track_number = as.numeric(track_number)
```

```
        ) |>
  filter(!is.na(track_popularity)))

mean_val <- mean(dat$track_popularity, na.rm = TRUE)
median_val <- median(dat$track_popularity, na.rm = TRUE)
q1_val <- quantile(dat$track_popularity, 0.25, na.rm = TRUE)
q3_val <- quantile(dat$track_popularity, 0.75, na.rm = TRUE)

ggplot(dat, aes(x = track_popularity)) +
  geom_histogram(bins = 30, alpha = 0.6, fill = "steelblue", color = "white") +
  geom_vline(aes(xintercept = mean_val, color = "Mean"), linewidth = 1.2, linetype = "solid") +
  geom_vline(aes(xintercept = median_val, color = "Median"), linewidth = 1.2, linetype = "dashed"
  geom_vline(aes(xintercept = q1_val, color = "Q1"), linewidth = 1.2, linetype = "dotted") +
  geom_vline(aes(xintercept = q3_val, color = "Q3"), linewidth = 1.2, linetype = "dotted") +
  annotate("text", x = mean_val, y = Inf, label = paste0("Mean: ", round(mean_val, 1)),
           vjust = 1.5, hjust = -0.1, color = "red", size = 4, fontface = "bold") +
  annotate("text", x = median_val, y = Inf, label = paste0("Median: ", round(median_val, 1)),
           vjust = 3, hjust = -0.1, color = "blue", size = 4, fontface = "bold") +
  annotate("text", x = q1_val, y = Inf, label = paste0("Q1: ", round(q1_val, 1)),
           vjust = 4.5, hjust = -0.1, color = "darkgreen", size = 4, fontface = "bold") +
  annotate("text", x = q3_val, y = Inf, label = paste0("Q3: ", round(q3_val, 1)),
           vjust = 6, hjust = -0.1, color = "darkgreen", size = 4, fontface = "bold") +
  scale_color_manual(values = c("Mean" = "red", "Median" = "blue", "Q1" = "darkgreen", "Q3" = "da
  scale_x_continuous(breaks = seq(0, 100, 10)) +
  labs(title = "Popularity Distribution with Statistics",
       x = "Popularity Score",
       y = "Count",
       color = "Statistics") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
        axis.title = element_text(size = 12))
```

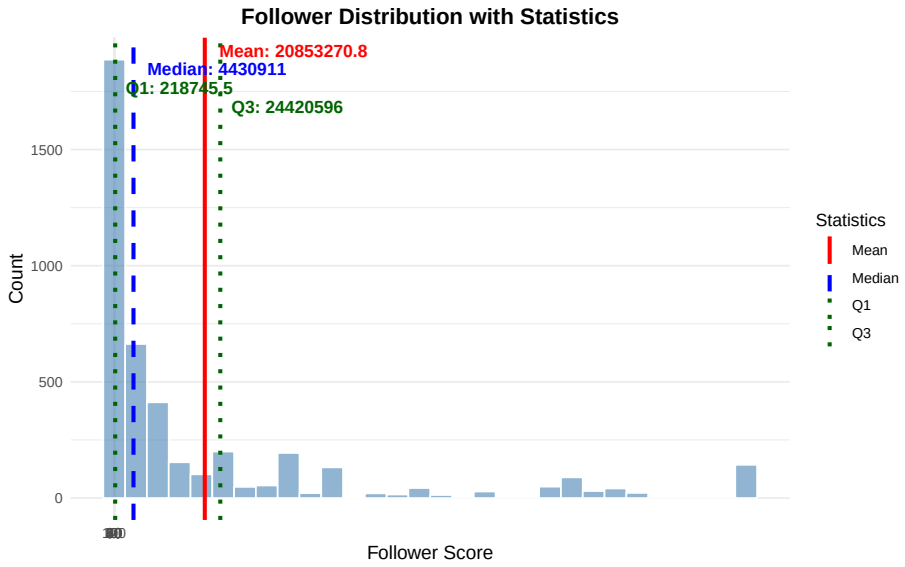



```r
library(ggplot2)
library(dplyr)

mean_val <- mean(dat$artist_followers, na.rm = TRUE)
median_val <- median(dat$artist_followers, na.rm = TRUE)
q1_val <- quantile(dat$artist_followers, 0.25, na.rm = TRUE)
q3_val <- quantile(dat$artist_followers, 0.75, na.rm = TRUE)

ggplot(dat, aes(x = artist_followers)) +
  geom_histogram(bins = 30, alpha = 0.6, fill = "steelblue", color = "white") +
  geom_vline(aes(xintercept = mean_val, color = "Mean"), linewidth = 1.2, linetype = "s
  geom_vline(aes(xintercept = median_val, color = "Median"), linewidth = 1.2, linetype
  geom_vline(aes(xintercept = q1_val, color = "Q1"), linewidth = 1.2, linetype = "dotte
  geom_vline(aes(xintercept = q3_val, color = "Q3"), linewidth = 1.2, linetype = "dotte
  annotate("text", x = mean_val, y = Inf, label = paste0("Mean: ", round(mean_val, 1))
           vjust = 1.5, hjust = -0.1, color = "red", size = 4, fontface = "bold") +
  annotate("text", x = median_val, y = Inf, label = paste0("Median: ", round(median_val
           vjust = 3, hjust = -0.1, color = "blue", size = 4, fontface = "bold") +
  annotate("text", x = q1_val, y = Inf, label = paste0("Q1: ", round(q1_val, 1)),
           vjust = 4.5, hjust = -0.1, color = "darkgreen", size = 4, fontface = "bold"
  annotate("text", x = q3_val, y = Inf, label = paste0("Q3: ", round(q3_val, 1)),
           vjust = 6, hjust = -0.1, color = "darkgreen", size = 4, fontface = "bold") +
  scale_color_manual(values = c("Mean" = "red", "Median" = "blue", "Q1" = "darkgreen",
  scale_x_continuous(breaks = seq(0, 100, 10)) +
  labs(title = "Follower Distribution with Statistics",
       x = "Follower Score",
       y = "Count",
```

```
      color = "Statistics") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
        axis.title = element_text(size = 12))
```




```
lm(track_popularity ~ artist_followers + explicit + as.numeric(track_number), data = dat) |> summ
```

```
Call:
lm(formula = track_popularity ~ artist_followers + explicit +
    as.numeric(track_number), data = dat)

Residuals:
    Min      1Q  Median      3Q     Max
-70.927 -12.532   3.163  17.276  50.417

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)               4.892e+01  5.158e-01  94.832  < 2e-
16 ***
artist_followers          1.606e-07  1.003e-08  16.006  < 2e-
16 ***
explicit                  4.744e+00  8.155e-01   5.817 6.41e-
09 ***
as.numeric(track_number) -2.259e-01  6.169e-02  -3.662 0.000253 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 22.68 on 4362 degrees of freedom
  (1 observation deleted due to missingness)
Multiple R-squared:  0.06962,   Adjusted R-squared:  0.06898
F-statistic: 108.8 on 3 and 4362 DF,  p-value: < 2.2e-16
```

**I regularly use `|>` known as a pipe operator. Values to the left are passed to the right. For instance, say you have a variable in your `R` script' where you subtract the mean then sample 100 values from the data.**

**The native R pipe operator is: `|>`, and there is another in `tidyr`, `%>%`. Both work similarly. If we want to apply a function to data,**

```
data |> function()
```

This is equivalent to:

```
function(data)
```

**The problem is code gets unreadable if we apply nested functions**

```
function1(function2(data))
```

**The pipe operator allows us to write this in a more readable manner:**

```
data |> function1() |>
  function2()
```

In this example, $b_0 = 48.92$, $b_{artist\_followers} = 1.61 \times 10^{-7}$, $b_{explicit} = 4.74$, and $b_{track\_number} = -0.23$. The substantive interpretation follows.

Artists that have more followers and release tracks with explicit content are more likely to have popular tracks. For `track_number` the coefficient is negative, indicating that tracks released earlier in an album are more popular; tracks released later on the album tend to be less popular. The linear regression estimates also allow for hypothesis tests. The $t - values$ $(b/se)$ amd associated p-values provide evidence suggesting that we might reject the null hypothesis $H_0 : b_i = 0$ for all three predictors.

These coefficients can also be interpreted directly. For every *unit* increase in $x$ there is a $b$ unit change in $y$. Let's consider the `explicit` variable. Here the coefficient is 4.74. Let's piece this together. Let's predict the average popularity at the average level of followers ($\bar{x} = 16.85$). This value is logged, so the average number of followers is $e^{16.85} \approx 20790361$. Let's fix track number at 2.

```
x = expand.grid(
  artist_followers = 16.85,
  explicit = c(0, 1),
  track_number = 2
)
lm(track_popularity ~ artist_followers + explicit + as.numeric(track_number), data = da
  predict(newdata = x)
```

```
       1        2
48.46481 53.20891
```

The difference between these two predictions is -2.64. The coefficient indicates what happens changing explicit *one unit*. The variable is binary, so *one-unit* simply means the effect of going from non-explicit to explicit content.

## 1.3 Anatomy of `lm`

The `lm()` function in R is the workhorse for fitting linear models. Understanding its structure and components is essential for working effectively with regression analysis in R.

### 1.3.1 Basic Syntax

The syntax is largely interchangeable with other statistical models in R: `lm(formula, data, ...)`.

The **formula** always follows the form `dv ~ iv1 + iv2 + ...`, where: - `dv` is the dependent variable (outcome) - `iv` are independent variables (predictors) - `~` means "is modeled as" or "is a function of" - `+` adds additional predictors

The **data** argument specifies the data frame containing the variables.

### 1.3.2 Components of an `lm` Object

When you run `lm()` and save the output to an object, that object – an `lm` object – contains numerous components that store different aspects of the regression analysis. You can explore these with `names()` or `ls()` and access them with `$`.

```
# Fit a model and save it
model <- lm(track_popularity ~ artist_followers + explicit + as.numeric(track_number), data = dat

# List all components
cat("Components of the lm object:\n")
```

```
Components of the lm object:
```

```
names(model)
```

```
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "na.action"     "xlevels"       "call"          "terms"
[13] "model"
```

Let's explore the most important components:

#### 1.3.2.1 1. Coefficients ($coefficients)

The estimated regression coefficients (intercept and slopes):

```
cat("Model coefficients:\n")
```

```
Model coefficients:
```

```
model$coefficients
```

```
             (Intercept)         artist_followers                 explicit
            4.891663e+01             1.606040e-07             4.744094e+00
as.numeric(track_number)
           -2.259099e-01
```

#### 1.3.2.2 2. Residuals ($residuals)

The differences between observed and fitted values ($e_i = y_i - \hat{y}_i$):

```
# First 10 residuals
head(model$residuals, 10)
```

```
        1         2         3         4         5         6         7         8
-53.20884 -53.81439 -49.46586 -22.30534 -53.21030 -46.69188 -
28.69099 -21.69214
        9        10
-32.69080 -20.45968
```

```
# Summary statistics of residuals
summary(model$residuals)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-70.927 -12.532   3.163   0.000  17.276  50.417
```

Key property: residuals should sum to (approximately) zero:

```
cat("Sum of residuals:", sum(model$residuals), "\n")
```

```
Sum of residuals: 6.970424e-12
```

#### 1.3.2.3 3. Fitted Values ($fitted.values)

The predicted values ($\hat{y}_i = a + b_1 x_{1i} + b_2 x_{2i} + ...$):

```
# First 10 fitted values
head(model$fitted.values, 10)
```

```
       1        2        3        4        5        6        7        8
53.20884 53.81439 53.46586 52.30534 53.21030 48.69188 48.69099 48.69214
       9       10
48.69080 53.45968
```

```
# Compare observed vs fitted
comparison <- data.frame(
  observed = dat$track_popularity[1:5],
  fitted = model$fitted.values[1:5],
  residual = model$residuals[1:5]
)
print(comparison)
```

```
  observed   fitted  residual
1        0 53.20884 -53.20884
2        0 53.81439 -53.81439
3        4 53.46586 -49.46586
4       30 52.30534 -22.30534
5        0 53.21030 -53.21030
```

#### 1.3.2.4 4. Model Matrix (`$model`)

The original data used in fitting, with the dependent variable and all predictors:

```
# First few rows of model matrix
head(model$model, 5)
```

```
  track_popularity artist_followers explicit as.numeric(track_number)
1                0          2812821        1                        4
2                0          2363438        1                        1
3                4           193302        1                        1
4               30          2813710        1                        8
5                0             8682        1                        2
```

#### 1.3.2.5 5. Call (`$call`)

The actual function call that created the model:

```
model$call
```

```
lm(formula = track_popularity ~ artist_followers + explicit +
    as.numeric(track_number), data = dat)
```

#### 1.3.2.6 6. Degrees of Freedom (`$df.residual`)

Residual degrees of freedom ($n - k$, where $n$ is sample size and $k$ is number of parameters):

```
cat("Residual degrees of freedom:", model$df.residual, "\n")
```

```
Residual degrees of freedom: 4362
```

```
cat("Sample size:", nobs(model), "\n")
```

```
Sample size: 4366
```

```
cat("Number of parameters:", length(model$coefficients), "\n")
```

```
Number of parameters: 4
```

### 1.3.3 Extracting More Information with Helper Functions

Beyond the direct components, several functions extract useful information:

#### 1.3.3.1 `summary()` - Comprehensive Model Summary

```
summary(model)
```

```
Call:
lm(formula = track_popularity ~ artist_followers + explicit +
    as.numeric(track_number), data = dat)

Residuals:
    Min      1Q  Median      3Q     Max
-70.927 -12.532   3.163  17.276  50.417

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)               4.892e+01  5.158e-01  94.832  < 2e-
16 ***
artist_followers          1.606e-07  1.003e-08  16.006  < 2e-
16 ***
explicit                  4.744e+00  8.155e-01   5.817 6.41e-
09 ***
as.numeric(track_number) -2.259e-01  6.169e-02  -3.662 0.000253 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 22.68 on 4362 degrees of freedom
  (1 observation deleted due to missingness)
Multiple R-squared:  0.06962,   Adjusted R-squared:  0.06898
F-statistic: 108.8 on 3 and 4362 DF,  p-value: < 2.2e-16
```

The summary includes: - Coefficient estimates with standard errors - t-statistics and p-values - $R^2$ and Adjusted $R^2$ - F-statistic for overall model significance - Residual standard error

#### 1.3.3.2 `coef()` - Extract Coefficients

```
coef(model)  # Same as model$coefficients
```

```
             (Intercept)         artist_followers                 explicit
```

```
            4.891663e+01              1.606040e-07              4.744094e+00
as.numeric(track_number)
           -2.259099e-01
```

#### 1.3.3.3 confint() - Confidence Intervals

```
# 95% confidence intervals for coefficients
confint(model, level = 0.95)
```

```
                                2.5 %        97.5 %
(Intercept)                4.790536e+01  4.992791e+01
artist_followers           1.409326e-07  1.802754e-07
explicit                   3.145266e+00  6.342922e+00
as.numeric(track_number) -3.468485e-01 -1.049712e-01
```

#### 1.3.3.4 vcov() - Variance-Covariance Matrix

```
vcov(model)
```

```
                          (Intercept) artist_followers      explicit
(Intercept)               2.660730e-01    -9.948830e-10 -1.421026e-
01
artist_followers         -9.948830e-10     1.006778e-16 -1.257256e-
09
explicit                 -1.421026e-01    -1.257256e-09  6.650682e-
01
as.numeric(track_number) -1.756984e-02    -1.511577e-10  1.858459e-
03
                         as.numeric(track_number)
(Intercept)                           -1.756984e-02
artist_followers                      -1.511577e-10
explicit                               1.858459e-03
as.numeric(track_number)               3.805333e-03
```

This matrix shows the variances (diagonal) and covariances (off-diagonal) of the coefficient estimates.

#### 1.3.3.5 residuals() and fitted() - Alternative Extraction

```
# These are equivalent to model$residuals and model$fitted.values
head(residuals(model), 3)
```

```
        1         2         3
-53.20884 -53.81439 -49.46586
```

```
head(fitted(model), 3)
```

```
       1        2        3
53.20884 53.81439 53.46586
```

#### 1.3.3.6 `predict()` - Generate Predictions

**`lm` can be paired with `predict()`. After estimation, simply provide `newdata` that contains the same predictor variables to generate predictions.**

```
# Predict for new data
new_tracks <- data.frame(
  artist_followers = c(16, 17, 18),
  explicit = c(0, 1, 0),
  track_number = c(1, 2, 3)
)

predictions <- predict(model, newdata = new_tracks)
cat("Predictions for new tracks:\n")
```

```
Predictions for new tracks:
```

```
print(predictions)
```

```
       1        2        3
48.69072 53.20891 48.23890
```

You can also get prediction intervals:

```
# Predictions with confidence and prediction intervals
predict(model, newdata = new_tracks, interval = "prediction", level = 0.95)
```

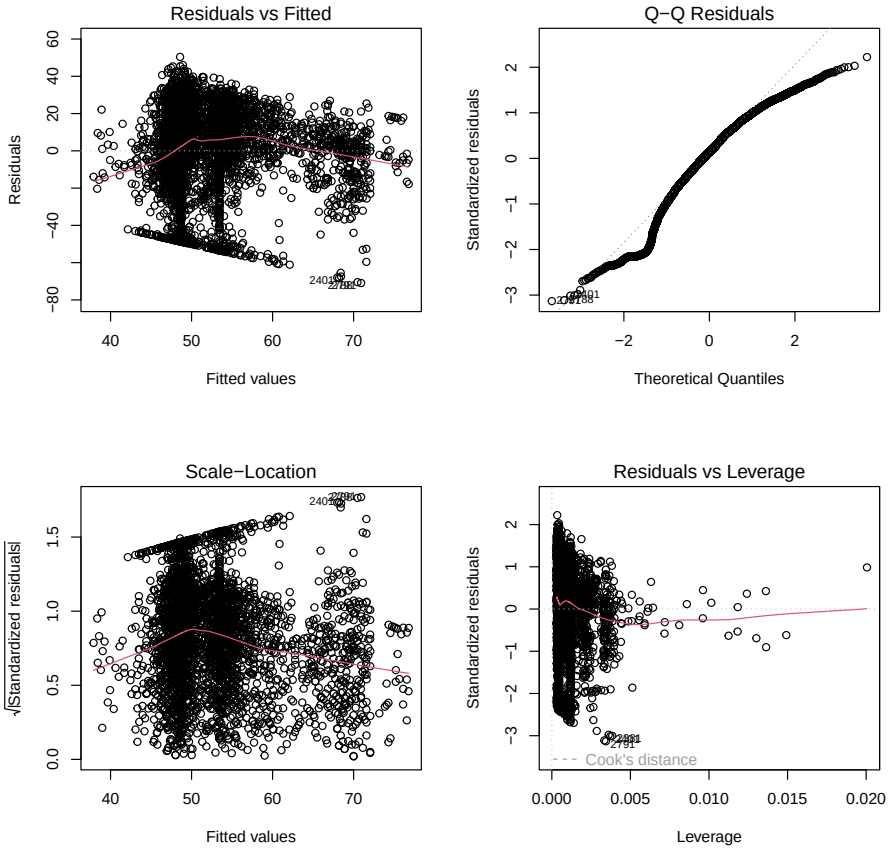
```
       fit      lwr      upr
1 48.69072 4.212357 93.16909
2 53.20891 8.714791 97.70303
3 48.23890 3.762259 92.71555
```

This decomposes the variation explained by each predictor sequentially.

### 1.3.4 Diagnostic Plots

The `plot()` function produces four diagnostic plots for model checking. These plots are essential for assessing whether the assumptions of linear regression are met. Let's examine each plot and what to look for:

```
par(mfrow = c(2, 2))
plot(model)
```




```
par(mfrow = c(1, 1))
```

#### 1.3.4.1 Plot 1: Residuals vs Fitted

**What the Plot Shows:** The residuals ($e_i = y_i - \hat{y}_i$) plotted against the fitted (predicted) values ($\hat{y}_i$).

1. **Linearity**: The points should be randomly scattered around the horizontal line at zero. If you see a pattern (e.g., curved, U-shaped), this suggests the relationship between X and Y may not be linear.
2. **Homoscedasticity (Constant Variance)**: The spread of residuals should be roughly constant across all fitted values. If the spread increases or decreases systematically (forming a funnel or cone shape), this indicates heteroscedasticity.
3. **Independence**: Points should not show systematic patterns. Any pattern suggests the model is missing something important.

#### 1.3.4.2 Plot 2: Normal Q-Q (Quantile-Quantile)

**What it shows:** Compares the standardized residuals to what we'd expect if they were normally distributed.

1. **Normality**: Points should fall approximately along the diagonal reference line. This line represents perfect normality. Just look for a 45 degree angle and line running from the lower left to upper right. Deviations from normality are observed from this line.
2. **Deviations**: Small deviations at the extremes are common and usually acceptable. Large deviations indicate non-normality.
3. **Heavy tails**: If points curve away from the line at both ends suggest heavy-tailed distributions (more extreme values than expected).
4. **Skewness**: Points systematically above or below the line suggest skewed distributions.

#### 1.3.4.3 Plot 3: Scale Location

**What it shows:** The square root of standardized residuals plotted against fitted values. This is another way to check for homoscedasticity.

1. **Constant variance**: The red smoothed line should be approximately horizontal, and the spread of points should be constant.
2. **Heteroscedasticity detection**: An upward or downward trend in the red line, or increasing/decreasing spread, indicates heteroscedasticity.

#### 1.3.4.4 Plot 4: Influence

**What it shows:** Standardized residuals plotted against leverage, with Cook's distance contours. This identifies influential observations.

**Key concepts:**

1. **Leverage** measures how far an observation's predictor values are from the mean. High leverage points have unusual X values.
2. **Residuals** measure how far an observation's Y value is from the predicted value. Large residuals indicate unusual Y values.
3. **Cook's Distance** combines leverage and residuals to measure overall influence. Points outside the Cook's distance contours (dashed lines) are highly influential.

## 1.4 Breaking Things Down

A statistical model – say linear equation – can be used to **estimate** the relationships between variables in a data set. It can be used *descriptively* to quantify

the strength of a relationship between two variables, perhaps controlling for additional variables. The procedure can also be extended *inferentially* to test hypotheses regarding inferences between *sample data* and the *population* from which the data are drawn.

*Always* consider the data generating process (DGP). Write a mathematical model, perhaps representing some process process. For instance, $y = \alpha + \beta x$. For every unit change in x, we observe a $\beta$ unit change in $y$. In this case, $\alpha$ is just a constant, representing the point at which the linear equation crosses the y axis, or equivalently, the value of $y$ when $x$ is zero.

In some cases, $x$ can be manipulated. We know the process *generating* different values of $x$. In an experiment – say a drug trial – $x$ is assigned (in the form of assignment to the treatment or control, and/or levels of the control. If we do not know, or purport to know, the factors that lead to $x$, we would say the data are observational.

Linear regression can be used with observational or experimental data. The method is the same; the interpretation of results are qualitatively different. With experimental data, the regression parameters can be used to generate *causal* estimates. With observational data, this is harder. It is possible, however, to estimate the magnitude of relationships between observational data; and in some cases, pair additional methods with regression to estimate causal parameters.

For instance, let's call a manipulated independent variable, $T_i$, representing assignment to a treatment condition. And, $y_i = \alpha + \beta T_i + e_i$. Here, $\beta$ is also known as the treatment effect, and may form the *Treatment Effect*.

# Chapter 2

# Types of Data, Types of Variables, Housekeeping

Let's first consider the difference between variables and constants. A random/stochastic variable is a variable that takes on a set of values and is assumed to be drawn from a distribution. We often make assumptions about these distributions. You've already learned about quite a few – the Gaussian (Normal), Gamma, Binomial, Poisson, etc. In the regression model, we will make *parameteric* assumptions about the distribution of errors. For much of the term, we'll assume normally distributed errors. A *constant* consists of those factors that do not vary across subjects – i.e., are not drawn from an underlying distribution. In the model,

$y_i = \alpha + \beta x_i$

The relationship between $x$ and $y$ is *perfect*, it is *deterministic*. In the social sciences, deterministic relationships are rare. Instead,

$y_i = \alpha + \beta x_i + \epsilon_i$

It is also important to carefully consider the structure of data. Simply visualizing the dataset and truly understanding the "unit of observation" will take you a long way in understanding statistical models.

It is also essential to understand the data structure, and the *unit of analysis*. In actual data, what does a row represent? Is it a person, a country, a state, a city, a region, a product, a survey response, or something else? Cross sectional data are data in which units are observed once. In a model, there are $i$ through $n$ observations of x, thus $x_i$. We can contrast this to time series data, where we observe a unit multiple, sometimes many times. For instance, a time series dataset might consist of quarterly unemployment data osberved over many years, in the United States. In time series a unit is observed more than once. We

typically index the time point with a **t**, so $x_t$. Each row is a time point.

Panel data consist of data in which a number of units are observed more than once. Now use **i** and **t**, so $x_{it}$. An example of a two-wave panel dataset is vote choice observed in both 2016 an 2020 among a group of survey respondents. A three-way panel consists of three waves, four-wave, four waves, and so on.

Understanding the structure of the data is essential to correctly specifying a statistical model. If our data are time series, fitting a regression model like, $y_t$=$\alpha$+$\beta x_t + \epsilon_t$ produces problems, formally **bias**. The reason is that the errors are likely correlated $cor(x_t, x_{t-1}) \neq 0$. As we'll see in this class, this violates one of the key assumptions of linear regression, that the errors are uncorrelated. Similar problems arise with panel data, but this *temporal autocorrelation* is less problematic if the data are cross sectional.

In addition to understanding the structure of the data, it is also necessary to understand the *nature* of the variables that constitute the **data**. **Ratio data** have a natural zero point, meaningful distances, and a natural ordering. Interval data have no natural zero point, but meaningful distances, and a natural ordering. These are the data types **for the dependent variable** which are generally implied in the linear regression model. In addition, ordinal data have no natural zero point, non-meaningful distances, but a meaningful ordering. Nominal data have no natural zero point, non-meaningful distances, and no natural ordering. In theory, ratio measures provide the most information about a variable; nominal measures, the least. All variable types are permissable as independent variables in a linear regression. However, if the dependent variable is ordinal, nominal, or binary, alternative models – many of which are explored in POL 683 – are more appropriate.

It's also common to call ratio and interval data "qualitative" data, and the remaining are "qualitative" data. Or, "continuous" versus "categorical."

## 2.1 Deriving the OLS Estimator

At the heart of linear regression is a linear equation.

First, assume $y$=$\alpha$+$\beta x$, so:

$$\begin{bmatrix} y_1 = & \alpha + \beta x_1 \\ y_2 = & \alpha + \beta x_2 \\ y_3 = & \alpha + \beta x_3 \\ y_4 = & \alpha + \beta x_4 \\ \vdots & \\ y_n = & \alpha + \beta x_n \end{bmatrix}$$

We could write this in a more compact manner as:

$$y_i = \alpha + \beta x_i$$

For the $i$th unit in the data set, we can predict that unit's $y$ value by plugging $x$ into the function. So $\alpha + \beta x_i$**.

There are two variables and two constants in this equation. There is one independent variable and one dependent variable. Likewise, we can say there is one endogeneous variable (y), and one exogenous variable (x). These are terms which will soon make more sense. Another way to write the equation is matrix and vector notation. $\mathbf{y} = \mathbf{X}\beta +$

## 2.2 Deterministic versus Stochastic

Most relationships in the social sciences are stochastic, not deterministic.

$$\begin{bmatrix} y_1 = & \alpha + \beta x_1 & +\epsilon_1 \\ y_2 = & \alpha + \beta x_2 & +\epsilon_2 \\ y_3 = & \alpha + \beta x_3 & +\epsilon_3 \\ y_4 = & \alpha + \beta x_4 & +\epsilon_4 \\ \vdots & & \\ y_n = & \alpha + \beta x_n & +\epsilon_n \end{bmatrix}$$

And,

$$\mathbf{y} = \mathbf{X}\beta + \mathbf{e}$$

# Chapter 3

# The Gauss-Markov Assumptions

This chapter is a bit more technical. It covers the assumptions – known as the Gauss-Markov assumptions – that underlie the Ordinary Least Squares (OLS) estimator. We'll explore these from both conceptual and mathematical perspectives.

First, recall that we rarely observe the PRF, $Y_i = \alpha + \beta_1 X_1 + \epsilon_i$ (the exception being a census) Instead, we observe the SRF, $Y_i = a + b_1 X_1 + e_i$.

We're then likely to use these values for $a$ and $b$ as point estimates of $\alpha$ and $\beta$

But because our estimates are subject to sampling error, point estimates should accompany an indicator uncertainty. We can estimate $Var(b)$ and $Var(a)$, allowing us to calculate standard errors, t-statistics, and confidence intervals. We can then use these to draw inferences.

If we make some *assumptions* about the PRF, the OLS estimator has several desirable properties. Some assumptions are critical for *estimation* others are critical for *inference*

**Assumption 1: Linearity.** The PRF is linear in parameters. *This assumption is necessary for estimation and is used to demonstrate unbiasedness.*

**Assumption 2: The IVs are exogenous and fixed.** The Xs are "given" and not determined within the model. The Xs are uncorrelated with the error term. $cov(X_i, \epsilon_i) = 0$.*This assumption is necessary for estimation and is used to demonstrate unbiasedness.*

**Assumption 3: The error has a zero mean.** Regardless of $X_i$, $E(\epsilon_i) = 0$. *This assumption is necessary for inference.*

$$E(\epsilon_i|X_i) = 0$$
$$E(\epsilon_i) = 0$$

There is no systematic effect of the error on our predicted values of $Y_i$, i.e., $\widehat{Y}_i$

**Assumption 4: Equal variance (homoskedasticity).** Regardless of $X_i$, $var(\epsilon_i) = \sigma^2$. *This assumption is necessary for inference.*

$$var(\epsilon_i|X_i) = \sigma^2, \quad \forall i$$
$$var(\epsilon_i) = \sigma^2, \quad \forall i$$

The variance around $\widehat{Y}_i$ is the same across levels of $X_i$.

**Assumption 5: Normality of error.** $\epsilon_i \sim N(0, \sigma)$. *This assumption is necessary for inference.*

Then, $Y_i \sim N(\alpha + \beta X_i, \sigma)$.

Note, this is an extension of assumptions 3 and 4. If we assume normality, the OLS estimator has minimum variance among *all* unbiased estimators.

**Assumption 6: Independent Errors.** $cov(\epsilon_i, \epsilon_j) = 0, \quad \forall i \neq j$. This simply reads "The $i$th error term is uncorrelated with the $j$th error term, for all i not equal to j." *This assumption is necessary for inference.*

This is the no autocorrelation assumption. With the exception of the relationship between $Y$s determined by $X$, there is no residual relationship between $Y$ variables.

**Assumption 7: Variation in $X$.**. This assumption is necessary for *estimation.*

**Assumption 8: Correct Specification.** This boils down to: (1) Correct functional form, and (2) Correctly included IVs. This assumption is necessary for *estimation.*

**Assumption 9: No perfect multicollinearity (with multiple Xs).** Recall, the denominator is 0 if $r_{x_1,x_2} = 1$. This assumption is necessary for *estimation.*

**Under these assumptions, it can be shown that the estimators for $\alpha$ and $\beta$ have desirable properties.**

## 3.1 The Gauss Markov Theorem

The Gauss-Markov (GM) Theorem is essential to OLS. The theorem – developed from the aformentioned assumption – states, *The Ordinary Least Squares (OLS) estimator is the **B**est **L**inear **U**nbiased **E**stimator (OLS is **BLUE**) with minimum variance of all unbiased linear estimators.*

The observation that the OLS estimator is **BLUE** means something quite specific; it is a technical term. It's a commonly misinterpreted characteristic in linear

regression estimated using OLS. The reason: **Best** means something specific as a statistical characteristic, but **Best** does not mean ideal or advisable in all circumstances, *even when the GM assumptions hold.*

Let's see what **BLUE** means in the context of OLS. This is what we can **demonstrate** when the GM assumptions hold

- **Linearity**: The estimators are a linear function of $Y_i$.
- **Unbiasedness**: $E(a) = \alpha$ and $E(b_k) = \beta_k$.
- **Minimum Variance**: Of all linear unbiased estimators, the OLS estimator will have minimum $var(a)$ and $var(b)$. Under normality, the OLS estimator has the minimum variance of all unbiased estimators. This assumption is critical for **efficiency**—the estimators have minimum sampling variance and smallest error.

Given the aforementioned assumptions, the OLS estimator falls in a class of linear estimators and is the best linear unbiased estimator of the intercept and slope parameter(s). **Why?**

$$\begin{bmatrix} y_1 = & b_0 + b_1 x_1 + e_1 \\ y_2 = & b_0 + b_1 x_2 + e_2 \\ y_3 = & b_0 + b_1 x_3 + e_3 \\ y_4 = & b_0 + b_1 x_4 + e_4 \\ \vdots & \\ y_n = & b_0 + b_1 x_n + e_n \end{bmatrix}$$

This is an important assumption – for each $y_i$ (the endogenous variable), it is a composite of the systematic component $(b_0 + b_1 x_i)$ and the unsystematic component $(e_i)$. The *unsystematic* component is simply the variation in $Y$ not explained by $X$.

**For the slope $b$:**

$$\begin{aligned} b &= \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sum(X_i - \bar{X})^2} \\ &= \frac{\sum(X_i - \bar{X})Y_i - \bar{Y}\sum(X_i - \bar{X})}{\sum(X_i - \bar{X})^2} \\ &= \frac{\sum(X_i - \bar{X})Y_i}{\sum(X_i - \bar{X})^2} \\ b &= \sum k_i Y_i, \text{ where} \\ k_i &= \frac{(X_i - \bar{X})}{\sum(X_i - \bar{X})^2} \end{aligned}$$

Let's pause a moment to consider this. First, we express the slope as the covariance between $X$ and $Y$ divided by the variance of $X$. If our variables were

standardized, this would be the correlation between $X$ and $Y$.

Notice that $\bar{Y}$ disappears in the second-to-third step. Recall that deviations from the mean always sum to zero. Now, just define $k_i$ as the parts of the slope that depend on $X_i$. Values further from $\bar{X}$ will have larger absolute values of $k_i$. They contribute to the slope. This should make some sense. Observations near the mean contribute little information to the steepness of the line.

**The OLS estimator $b$ is a linear function of $Y_i$ with weights $k_i$.**

### 3.1.1 Properties of $k_i$

The weights $k_i$ have four important properties:

| Property | Statement | Why it matters |
|---|---|---|
| 1. | $\sum k_i = 0$ | Eliminates $\alpha$ when proving $E(b) = \beta$ |
| 2. | $\sum k_i X_i = 1$ | Ensures the coefficient on $\beta$ equals 1 |
| 3. | $\sum k_i^2 = \frac{1}{\sum (X_i - \bar{X})^2}$ | Used to derive $var(b)$ |
| 4. | $k_i$ are constants | Because $X$ is fixed (Assumption 2), we can factor $k_i$ out of expectations |

**1.** $\sum k_i = 0$

$$\sum k_i = \sum \frac{(X_i - \bar{X})}{\sum (X_j - \bar{X})^2} = \frac{\sum (X_i - \bar{X})}{\sum (X_j - \bar{X})^2} = \frac{0}{\sum (X_j - \bar{X})^2} = 0$$

The numerator equals zero because deviations from the mean always sum to zero: $\sum (X_i - \bar{X}) = \sum X_i - n\bar{X} = n\bar{X} - n\bar{X} = 0$.

**2.** $\sum k_i X_i = 1$

$$\sum k_i X_i = \frac{\sum (X_i - \bar{X}) X_i}{\sum (X_j - \bar{X})^2}$$

For the numerator, decompose $X_i = (X_i - \bar{X}) + \bar{X}$:

$$\begin{aligned} \sum (X_i - \bar{X}) X_i &= \sum (X_i - \bar{X}) \left[ (X_i - \bar{X}) + \bar{X} \right] \\ &= \sum (X_i - \bar{X})^2 + \bar{X} \underbrace{\sum (X_i - \bar{X})}_{=0} \\ &= \sum (X_i - \bar{X})^2 \end{aligned}$$

Therefore: $\sum k_i X_i = \frac{\sum (X_i - \bar{X})^2}{\sum (X_j - \bar{X})^2} = 1$

**3.** $\sum k_i^2 = \frac{1}{\sum (X_i - \bar{X})^2}$

$$\sum k_i^2 = \sum \left[ \frac{(X_i - \bar{X})}{\sum (X_j - \bar{X})^2} \right]^2 = \sum \frac{(X_i - \bar{X})^2}{\left[\sum (X_j - \bar{X})^2\right]^2} = \frac{\sum (X_i - \bar{X})^2}{\left[\sum (X_j - \bar{X})^2\right]^2} = \frac{1}{\sum (X_i - \bar{X})^2}$$

This property is crucial for deriving the variance of $b$.

#### 3.1.1.1 Simulation: Verifying the Properties of $k_i$

Let's verify these properties with simulated data. We'll draw $X$ and $Y$ from a bivariate normal distribution and compute $k_i$.

```
set.seed(42)

library(MASS)
n <- 100
mu <- c(5, 10)  # means of X and Y
Sigma <- matrix(c(4, 3,    # variance of X = 4, covariance = 3
                  3, 9),   # variance of Y = 9
                nrow = 2)

data <- mvrnorm(n, mu, Sigma)
X <- data[, 1]
Y <- data[, 2]
head(data)
```

```
         [,1]      [,2]
[1,] 5.123798 14.825352
[2,] 2.703320  9.064260
[3,] 6.960291 10.375307
[4,] 3.169354 13.111677
[5,] 6.525794 10.725355
[6,] 4.700879  9.762086
```

```
lm(Y ~ X)
```

```
Call:
lm(formula = Y ~ X)

Coefficients:
(Intercept)            X
     5.1900       0.9368
```

```r
# Compute k_i weights
x_dev <- X - mean(X)
SS_x <- sum(x_dev^2)
k <- x_dev / SS_x

# Verify Property 1: sum(k_i) = 0
cat("Property 1: sum(k_i) =\n", sum(k), "\n")
```

```
Property 1: sum(k_i) =
 -1.292369e-16
```

```r
# Verify Property 2: sum(k_i * X_i) = 1
cat("Property 2: sum(k_i * X_i) =", sum(k * X), "\n")
```

```
Property 2: sum(k_i * X_i) = 1
```

```r
# Verify Property 3: sum(k_i^2) = 1 / sum(x_i^2)
cat("Property 3: sum(k_i^2) =", sum(k^2), "\n")
```

```
Property 3: sum(k_i^2) = 0.002765756
```

```r
cat("            1/sum(x_i^2) =", 1/SS_x, "\n")
```

```
            1/sum(x_i^2) = 0.002765756
```

```r
# Bonus: Verify that b = sum(k_i * Y_i) matches lm()
b_manual <- sum(k * Y)
b_lm <- coef(lm(Y ~ X))[2]
cat("\nSlope from sum(k_i * Y_i):", b_manual, "\n")
```

```
Slope from sum(k_i * Y_i): 0.9367911
```

```r
cat("Slope from lm():", b_lm, "\n")
```

```
Slope from lm(): 0.9367911
```

The simulation confirms:

- $\sum k_i \approx 0$ (machine precision)
- $\sum k_i X_i = 1$ exactly
- $\sum k_i^2 = 1/\sum (X_i - \bar{X})^2$ exactly
- Computing $b = \sum k_i Y_i$ gives the same slope as `lm()`

**Creating a Function for $k_i$ Weights**

In practice, you'll want to encapsulate the calculation of $k_i$ weights into a reusable function. Functions drastically increase code clarity, maintainability, and reusability. They also help with testing and productivity. Instead of cutting and pasting code snippets which do the same thing, write one function and just call that function.
A function means something particular to `R`. It is a block of code that performs a specific task, but can operate on different input. Here's how we might accomplish this for computing $k_i$ weights:

```
compute_k_weights <- function(X) {
  x_dev <- X - mean(X)
  SS_x <- sum(x_dev^2)
  k <- x_dev / SS_x
  return(k)
}
```

**Example Usage**

```
X <- c(2, 4, 6, 8, 10)
k <- compute_k_weights(X)

sum(k)            # Should be   0
sum(k * X)        # Should be 1
sum(k^2)          # Should be 1/sum((X - mean(X))^2)
```

**For the intercept $a$:**

Recall that $a = \bar{Y} - b\bar{X}$. Substituting $b = \sum k_i Y_i$:

$$
\begin{aligned}
a &= \bar{Y} - b\bar{X} \\
a &= \frac{1}{n}\sum Y_i - \bar{X}\sum k_i Y_i \\
a &= \sum \left(\frac{1}{n} - \bar{X}k_i\right) Y_i \\
a &= \sum c_i Y_i, \text{ where} \\
c_i &= \frac{1}{n} - \bar{X} \cdot \frac{(X_i - \bar{X})}{\sum (X_i - \bar{X})^2}
\end{aligned}
$$

**The OLS estimator $a$ is also a linear function of $Y_i$ with weights $c_i$.**

*These are important results, primarily because they make subsequent operations more tractable – like demonstrating unbiasedness and minimum variance. The reason, is that we can understand these parameters to include only $Y_i$ and constants (the weights).*

## 3.2 Unbiasedness

$b$ is an unbiased estimator of $\beta$.

**In words, the expected value of $b$ equals the population parameter $\beta$.**

**And maybe more intuitive:**, the average value of $b$ across repeated samples equals the population parameter $\beta$.

$$
\begin{aligned}
b &= \sum k_i Y_i \\
b &= \sum k_i(\alpha + \beta X_i + \epsilon_i) \\
b &= \alpha \sum k_i + \beta \sum k_i X_i + \sum k_i \epsilon_i \\
b &= \alpha \cdot 0 + \beta \cdot 1 + \sum k_i \epsilon_i \\
b &= \beta + \sum k_i \epsilon_i \\
E(b) &= \beta
\end{aligned}
$$

The intercept $\alpha$ vanishes because $\sum k_i = 0$. The coefficient on $\beta$ equals 1 because $\sum k_i X_i = 1$. These aren't coincidences—they're properties built into how $k_i$ is defined.

**Random Variables vs. Constants**

Understanding the distinction between random variables and constants is essential for working with expectations.

**A constant** is a fixed, known value that does not vary across samples or observations. Examples in our context:

- $\alpha$ and $\beta$ (the true population parameters)
- $X_i$ (treated as fixed by Assumption 2)
- $k_i$ (derived entirely from $X$ values)
- $\bar{X}$ (the mean of fixed $X$ values)

**A random variable** is a quantity whose value is determined by a random process—it varies across samples. Examples:

- $\epsilon_i$ (the error term—different in each sample)
- $Y_i$ (because $Y_i = \alpha + \beta X_i + \epsilon_i$ contains $\epsilon_i$)
- $b$ and $a$ (because they're functions of $Y_i$)

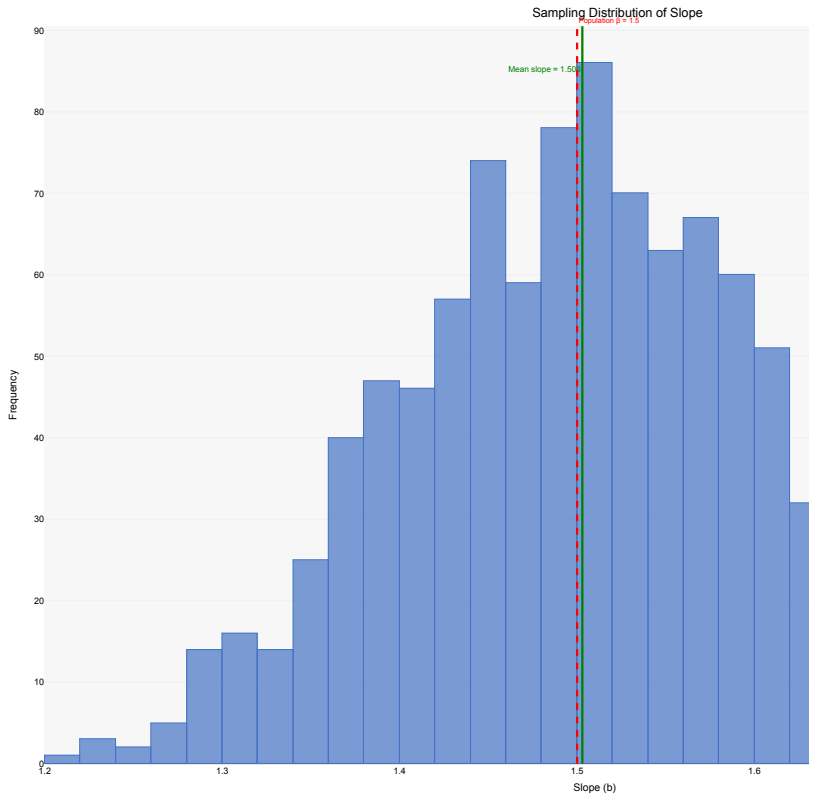
**Why this matters for expectations:**

- Constants can be factored out: $E(cX) = cE(X)$
- The expectation of a constant is itself: $E(c) = c$
- The expectation of a random variable is its long-run average

In our derivation, $k_i$ are constants, so $E(\sum k_i \epsilon_i) = \sum k_i E(\epsilon_i)$. We can pull $k_i$ outside the expectation because it doesn't vary—only $\epsilon_i$ does.

### 3.2.1 Simulation: Demonstrating Unbiasedness

Let's verify unbiasedness with a simulation. We'll draw repeated samples from a population with known parameters, calculate the slope for each sample, and observe that the average slope converges to the true population slope.




The simulation confirms unbiasedness: across many repeated samples, the average slope equals the true population parameter $\beta$. The red dashed line shows

the population slope, while the green line shows the sample mean—they align perfectly (within sampling variation).

**Properties of the expectation operator** used in this derivation:

1. **Linearity**: $E(aX + bY) = aE(X) + bE(Y)$. The expectation passes through sums and pulls out constants.
2. **Constants**: $E(c) = c$ — the expected value of a constant is that constant
3. **Scaling**: $E(cX) = cE(X)$ — constants factor out of expectations

But what about the variance? We've shown unbiasedness, but how much does $b$ vary from sample to sample?

**Definition of variance:**

$$var(b) = E\left[(b - E(b))^2\right]$$

The variance measures how spread out the sampling distribution of $b$ is around its expected value.

**Deriving $var(b)$:**

We established that $b = \beta + \sum k_i \epsilon_i$. Since $E(b) = \beta$:

$$b - E(b) = b - \beta = \sum k_i \epsilon_i$$

Therefore:

$$\begin{aligned} var(b) &= E\left[(b - \beta)^2\right] \\ &= E\left[(\sum_i k_i \epsilon_i)^2\right] \\ &= E\left[(\sum_i k_i \epsilon_i)(\sum_j k_j \epsilon_j)\right] \\ &= E\left[\sum_i \sum_j k_i k_j \epsilon_i \epsilon_j\right] \end{aligned}$$

The *inside* of the expectation is a double summation over all $i$ and $j$. It forms a matrix of terms. Each term is weighted by $k_i k_j$ and involves the product of two error terms, $\epsilon_i \epsilon_j$.

Now we can implement two assumptions **about the error terms**.

- **Assumption 4 (Homoskedasticity)**: $var(\epsilon_i) = \sigma^2$ for all $i$
- **Assumption 6 (Independence)**: $cov(\epsilon_i, \epsilon_j) = 0$ for $i \neq j$

**Properties of Covariance and Variance**

**Covariance** measures how two random variables move together:

$$cov(X,Y) = E[(X - E(X))(Y - E(Y))] = E(XY) - E(X)E(Y)$$

**Key rearrangement:** $E(XY) = cov(X,Y) + E(X)E(Y)$
This tells us how to evaluate the expected value of a product of two random variables.
**Variance** is the covariance of a variable with itself:

$$var(X) = cov(X,X) = E(X^2) - [E(X)]^2$$

**Key rearrangement:** $E(X^2) = var(X) + [E(X)]^2$
**Important properties:**

- If $X$ and $Y$ are independent, then $cov(X,Y) = 0$
- If $E(X) = 0$, then $E(X^2) = var(X)$
- If $X$ and $Y$ are independent with $E(X) = E(Y) = 0$, then $E(XY) = 0$

**Applying these properties to our matrix:**

The expectation of the double sum consists of variance terms on the diagonal ($i = j$) and covariance terms off the diagonal ($i \neq j$).

- **Off-diagonal** ($i \neq j$): $E(\epsilon_i \epsilon_j) = cov(\epsilon_i, \epsilon_j) + E(\epsilon_i)E(\epsilon_j) = 0 + 0 = 0$ (by A6 and A3)
- **Diagonal** ($i = j$): $E(\epsilon_i^2) = var(\epsilon_i) + [E(\epsilon_i)]^2 = \sigma^2 + 0 = \sigma^2$ (by A4 and A3)

So only the diagonal terms ($i = j$) remain:

$$\begin{aligned} var(b) &= E\left[\sum_i k_i^2 \epsilon_i^2\right] \\ &= \sum_i k_i^2 E(\epsilon_i^2) \\ &= \sum_i k_i^2 \cdot \sigma^2 \\ &= \sigma^2 \sum k_i^2 \end{aligned}$$

$$var(b) = \frac{\sigma^2}{\sum (X_i - \bar{X})^2}$$

This formula reveals what affects the *precision* of the slope estimates – how much do they vary around the true population parameter $\beta$?

Note that **larger** $\sigma^2$ indicates there is more noise in $y_i$. In this case, we expect $b$ to vary more from sample to sample. On the other hand, the more variance in $x$, the more precise our estimates of $b$. This relates to what we discussed earlier regarding the weights $k_i$. The more spread in $X_i$ the more they contribute to our knowledge of the slope; we can estimate the slope more precisely, i.e, with less variance. *We learn more about* $E(Y|X) = \beta$ *when* $X_i$ *varies.*

## 3.3 Minimum Variance

Consider what this means:

$$\begin{aligned} var(b) &= E(b - E(b))^2 \\ &= \frac{\sigma^2}{\sum x_i^2} \end{aligned}$$

The $var(b)$ will increase as the population variance increases.

The $var(b)$ will increase as the $x$ variance decreases.

But, there are two problems:

1. We rarely have access to $\sigma^2$;
2. We have not yet shown that $var(b)$ is the minimum variance.

It can be shown that $\hat{\sigma^2}$ is an unbiased estimator of $\sigma^2$ (i.e., $E(\hat{\sigma}^2) = \sigma^2$; you likely encountered this in 682). So, use $\hat{\sigma^2}$ in place of $\sigma^2$. Where,

$$\begin{aligned} \hat{\sigma^2} &= RSS/(n-k-1) \\ &= \sum e_i^2/(n-k-1) \end{aligned}$$

And,

$$\begin{aligned} var(b) &= \frac{\hat{\sigma^2}}{\sum x_i^2} \\ var(a) &= \frac{\hat{\sigma^2} \sum X_i^2}{n \sum x_i^2} \end{aligned}$$

To show minimum variance, let's generate an alternate estimate of $b$, call it $b^*$.

$$\begin{aligned} b &= \sum k_i Y_i \\ b^* &= \sum w_i Y_i \end{aligned}$$

If $b^*$ is unbiased, then $\sum w_i = 0$, and $\sum w_i X_i = 1$.

Even without knowing these weights, we can still operate as if they're something unknown, $w$:

$$var(b^*) = \sigma^2 \left( \sum w_i - \frac{x_i}{\sum x_i} \right) + \frac{\sigma^2}{\sum x_i^2}$$

But consider what this actually means:

$$var(b^*) = \sigma^2 \left( \sum w_i - \frac{x_i}{\sum x_i} \right) + \frac{\sigma^2}{\sum x_i^2}$$

Only if $w_i = k_i$ will we obtain the variance of the OLS estimator. There is no alternative estimate of the variance less than this value. And here ends our direct proof of minimum variance.

# Chapter 4

# Estimation in R

We'll generally try to use *real* data in this class. However, it is useful to generate our own *synthetic* data. Eventually we can build this into Monte Carlo Experiments to explore the characteristics of our estimators. I find this useful to understand how sampling, the DGP, coding and estimation all come together.

In the last chapter, we generated synthetic data using `R` functions. Let's extend that logic here.

**Data Generating Process (DGP)**: The **DGP** can be thought of as the *underlying* process that produces the (sample) data we observe. This is akin to the *PRF* versus *SRF* distinction we already discussed. But here, we're directly simulating data from a *known* process.

```
simulate_regression_data <- function(
  n = 500,                    # Sample size
  beta_0 = 0,                 # Intercept
  beta_1 = 0.2,               # Slope (controls correlation strength)
  x_mean = 0,                # Mean of X
  x_sd = 1,                   # Standard deviation of X (controls variance)
  error_sd = 1                # Standard deviation of normal
) {

  X <- rnorm(n, mean = x_mean, sd = x_sd)

  errors <- rnorm(n, mean = 0, sd = error_sd)

  Y <- beta_0 + beta_1 * X + errors

  # Return as data frame
  data.frame(
    x = X,
```

```
    y = Y,
    true_y = beta_0 + beta_1 * X,  # Y without error
    error = errors
  )
}
```

```
set.seed(123)
sim_dat <- simulate_regression_data(
  n = 500,
  beta_1 = 0.75,
  x_sd = 1,
  error_sd = 1
)
head(sim_dat)
```

```
           x          y      true_y       error
1 -0.56047565 -1.0222496 -0.42035673 -0.60189285
2 -0.23017749 -1.1663317 -0.17263312 -0.99369859
3  1.55870831  2.1958163  1.16903124  1.02678506
4  0.07050839  0.8039426  0.05288129  0.75106130
5  0.12928774 -1.4122007  0.09696580 -1.50916654
6  1.71506499  1.1911513  1.28629874 -0.09514745
```

```
library(ggplot2)

# Fit the sample regression line
fit <- lm(y ~ x, data = sim_dat)
sim_dat$fitted_y <- fitted(fit)

# Calculate R-squared
r_squared <- summary(fit)$r.squared

ggplot(sim_dat, aes(x = x, y = y)) +
  geom_segment(aes(xend = x, yend = fitted_y),
               alpha = 0.2, color = "gray50", linewidth = 0.3) +
  geom_point(alpha = 0.5, size = 2) +
  geom_line(aes(y = true_y), color = "#d62728", linewidth = 1.2) +
  geom_line(aes(y = fitted_y), color = "#2c7fb8", linewidth = 1.2, linetype = "dashed")
  annotate("text", x = min(sim_dat$x) + 1, y = max(sim_dat$y) - 1,
           label = paste0("R² = ", round(r_squared, 3)),
           size = 5, hjust = 0) +
  labs(
    title = "PRF vs. SRF: Understanding the Data Generating Process",
    subtitle = "Gray lines show residuals (deviations from the fitted line)",
    x = "X",
    y = "Y"
```

```
  ) +
  theme_minimal(base_size = 12) +
  theme(
    plot.title = element_text(face = "bold"),
    plot.subtitle = element_text(color = "gray40")
  )
```

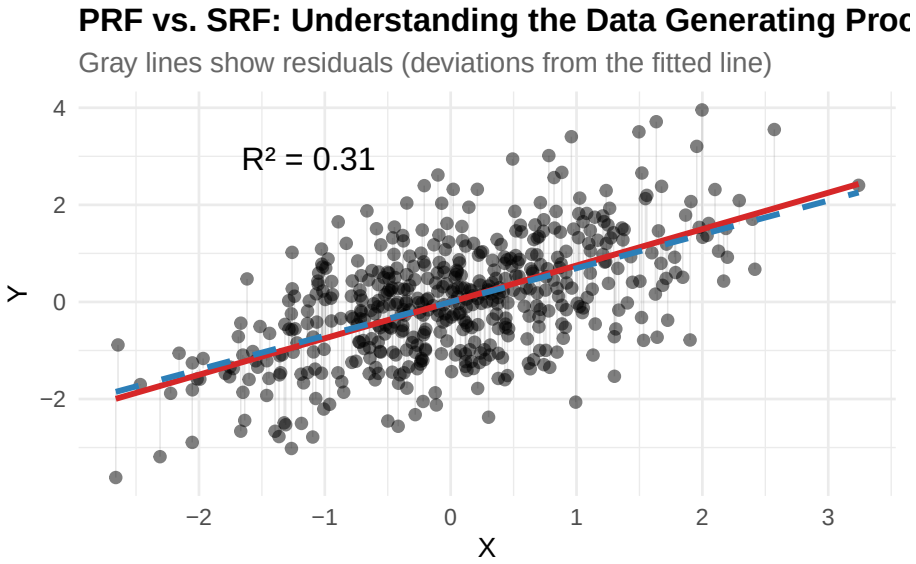



Figure 4.1: Comparing the Population Regression Function (PRF) to the Sample Regression Function (SRF)

The red solid line is the **PRF** (Population Regression Function): the true data generating process with $\beta_0 = 2$ and $\beta_1 = 0.8$. The blue dashed line is the **SRF** (Sample Regression Function): our estimated regression line from the observed data. The gray lines show the **residuals**—deviations from each observed point to the fitted line (SRF).

## 4.1 Estimation

```
# Fit the linear model
fit <- lm(y ~ x, data = sim_dat)
summary(fit)
```

```
Call:
lm(formula = y ~ x, data = sim_dat)
```

```
Residuals:
     Min       1Q   Median       3Q      Max
-2.75568 -0.67016  0.01042  0.63073  2.73709

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.0004678  0.0452219   -0.01    0.992
x            0.6960271  0.0465049   14.97   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.011 on 498 degrees of freedom
Multiple R-squared:  0.3103,    Adjusted R-squared:  0.3089
F-statistic:   224 on 1 and 498 DF,  p-value: < 2.2e-16
```

***Elements of the output:*** - **Coefficients**: The estimated intercept and slope, along with their standard errors, t-values, and p-values. - **Residual standard error**: The average distance that the observed values fall from the regression line - **R-squared**: The proportion of variance in the dependent variable that is explained by the independent variable(s). - **F-statistic**: Tests whether at least one predictor variable has a non-zero coefficient

**Generating Predictions**: We can use the fitted model to generate predictions for new data points. All we need to do is create a new **data frame** with the same structure as the original data (i.e., it should have a column named `x`), and then pass it to the `predict()` function.

```
cat("Predicted value for x = 1:",
predict(fit, newdata = data.frame(x = 1))
)
```

```
Predicted value for x = 1: 0.6955593
```

Likewise, we could generate predictions for a range of `x` values.

```
multiple_x <- seq(0.25, 0.35, by = 0.05)
predict(fit, newdata = data.frame(x = multiple_x))
```

```
        1         2         3
0.1735390 0.2083404 0.2431417
```

Or we could just use the original data points to create the fitted values – the predictions for each $x_i$ in the original data.

```
predict(fit) |>
head()
```

```
          1           2           3           4           5           6
-0.39057401 -0.16067754  1.08443545  0.04860798  0.08952000  1.19326394
```

## 4.2 Model Fit: Understanding $R^2$, Correlation, and ANOVA

Recall from earlier chapters that there's an intimate relationship between bivariate linear regression, the correlation coefficient, and *Analysis of Variance* (ANOVA).

The correlation $r$ is the same as the standardized slope coefficient in a bivariate regression.

```
fit <- lm(scale(y) ~ scale(x), data = sim_dat)
summary(fit)
```

```
Call:
lm(formula = scale(y) ~ scale(x), data = sim_dat)

Residuals:
     Min       1Q   Median       3Q      Max
-2.26700 -0.55132  0.00857  0.51888  2.25170

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 3.351e-17  3.718e-02    0.00        1
scale(x)    5.570e-01  3.722e-02   14.97   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8313 on 498 degrees of freedom
Multiple R-squared:  0.3103,    Adjusted R-squared:  0.3089
F-statistic:   224 on 1 and 498 DF,  p-value: < 2.2e-16
```

```
cor(sim_dat$x, sim_dat$y)
```

```
[1] 0.5570035
```

Because we've simulated these data, we also know the **true** data generating process. This gives us a unique opportunity to explore how well our sample regression function (SRF) recovers the parameters of the population regression function (PRF).

### 4.2.1 Decomposing the Variance

The fundamental decomposition in ANOVA is:

$$TSS = RegSS + RSS$$

Or equivalently:

$$\sum(Y_i - \bar{Y})^2 = \sum(\hat{Y}_i - \bar{Y})^2 + \sum(Y_i - \hat{Y}_i)^2$$

Where:

- $TSS = \sum(Y_i - \bar{Y})^2$ is the **Total Sum of Squares**. This is the total variation in $Y$ around its mean.
- $RegSS = \sum(\hat{Y}_i - \bar{Y})^2$ is the **Regression Sum of Squares**. This is the variation in the *fitted values* around the mean of $Y$. It represents the reduction in error from using the linear model rather than just $\bar{Y}$.
- $RSS = \sum(Y_i - \hat{Y}_i)^2$ is the **Residual Sum of Squares**. This is the variation in $Y$ that remains *unexplained* by the model.

Let's "manually" calculate these for our simulated data:

```
# Calculate the mean of y
y_bar <- mean(sim_dat$y)

# Calculate TSS, RegSS, and RSS
TSS <- sum((sim_dat$y - y_bar)^2)
RegSS <- sum((sim_dat$fitted_y - y_bar)^2)
RSS <- sum((sim_dat$y - sim_dat$fitted_y)^2)

# Display the decomposition
cat("Total Sum of Squares (TSS):", round(TSS, 3), "\n")
```

```
Total Sum of Squares (TSS): 737.321
```

```
cat("Regression Sum of Squares (RegSS):", round(RegSS, 2), "\n")
```

```
Regression Sum of Squares (RegSS): 228.76
```

```
cat("Residual Sum of Squares (RSS):", round(RSS, 3), "\n")
```

```
Residual Sum of Squares (RSS): 508.565
```

```
cat("RegSS + RSS =", round(RegSS + RSS, 3), "\n")
```

```
RegSS + RSS = 737.321
```

```
cat("Difference from TSS:", round(TSS - (RegSS + RSS), 3), "\n")
```

```
Difference from TSS: 0
```

This seems to work: $TSS = RegSS + RSS$. The difference is effectively zero…maybe a little off simply due to rounding.

### 4.2.2 $R^2$: The Coefficient of Determination

$R^2$ is defined as the *proportional reduction in error* from using the linear model. **How much better do we explain variation in $Y$ using our model? Does it explain more variation than just using the mean of $Y$?**

$$R^2 = \frac{RegSS}{TSS} = 1 - \frac{RSS}{TSS}$$

$R^2$ is a statistic that ranges from 0 to 1. An $R^2$ of 0 means the model explains *none* of the variation in $Y$. $E(Y|X) = E(Y) = \bar{Y}$

At the other extreme $R^2$ of 1 means the model explains *all* of the variation, a **deterministic* relationship; a correlation of 1.

```
# Calculate R-squared manually
r_squared_manual <- RegSS / TSS
r_squared_from_model <- summary(fit)$r.squared

cat("R² (manual calculation):", round(r_squared_manual, 4), "\n")
```

```
R² (manual calculation): 0.3103
```

```
cat("R² (from model summary):", round(r_squared_from_model, 4), "\n")
```

```
R² (from model summary): 0.3103
```

Both calculations yield the same result. Our model explains approximately 31% of the variation in $Y$.

### 4.2.3 The Relationship Between $R^2$ and Correlation

To bring things full circle, let's connect $R^2$ to the correlation coefficient, $r$. In *bivariate* regression – i.e., one predictor – there's a direct correspondence between $R^2$ and the Pearson correlation coefficient, $r$:

$$R^2 = r^2$$

More precisely:

$$r = \sqrt{R^2} \times \text{sign}(\beta)$$

Where sign($\beta$) is positive if the slope is positive, negative if the slope is negative. Let's verify this:

```
# Calculate correlation
correlation <- cor(sim_dat$x, sim_dat$y)
slope_sign <- sign(coef(fit)[2])
```

```
cat("Correlation (r):", round(correlation, 4), "\n")
```

```
Correlation (r): 0.557
```

```
cat("Square root of R²:", round(sqrt(r_squared_from_model), 4), "\n")
```

```
Square root of R²: 0.557
```

```
cat("Sign of slope:", slope_sign, "\n")
```

```
Sign of slope: 1
```

```
cat("√R² × sign( ):", round(sqrt(r_squared_from_model) * slope_sign, 4), "\n")
```

```
√R² × sign( ): 0.557
```

```
cat("\nVerification: r² = R²?", round(correlation^2, 4), "=", round(r_squared_from_mode
```

```
Verification: r² = R²? 0.3103 = 0.3103
```

The relationship again is apparent. $R^2$ is simply the squared correlation coefficient or the squared standardized coefficient in bivariate regression.

### 4.2.4 The F-statistic and ANOVA

The F-statistic tests the null hypothesis that *all* slope coefficients are zero. In bivariate regression, this is equivalent to testing whether $\beta = 0$. The F-statistic is calculated from the ANOVA decomposition:

$$F = \frac{RegSS/df_{reg}}{RSS/df_{res}} = \frac{MSS_{reg}}{MSS_{res}}$$

Where $df_{reg} = k$ (the number of predictors) and $df_{res} = n - k - 1$. The Mean Squares are the Sum of Squares divided by their degrees of freedom.

#### 4.2.4.1 Why the F-distribution?

Thinking back to POL 681, recall the F-statistic is a *ratio of two variances* (the mean squares). These variances are individually distributed as scaled chi-squared distributions under the null hypothesis.

In particular, *both* $RegSS/\sigma^2$ and $RSS/\sigma^2$ follow & ^2$ distributions (scaled by their respective degrees of freedom). The ratio of two independent chi-squared random variables, each divided by its degrees of freedom, follows an **F-distribution** with degrees of freedom $(df_{reg}, df_{res})$.

In our case, the F-distribution describes the distribution of the **ratio** of two estimated variances when the null hypothesis is true. If the model explains a lot of variance, the F-statistic will be large. If not, will approach zero. We use

F-tables or p-values to determine whether our observed F-statistic is unlikely under the null hypothesis.

Using null hypothesis testing, the hypotheses may be states as:

- $H_0$: $\beta = 0$ (the model does not explain more variance than the null model)
- $H_a$: $\beta \neq 0$ (the model explains more variance than the null model)

```
# Extract from model
n <- nobs(fit)
k <- 1  # one predictor

df_reg <- k
df_res <- n - k - 1

MSS_reg <- RegSS / df_reg
MSS_res <- RSS / df_res

F_stat <- MSS_reg / MSS_res

# Compare to model output
model_f <- summary(fit)$fstatistic[1]

cat("F-statistic (manual calculation):", round(F_stat, 4), "\n")
```

```
F-statistic (manual calculation): 224.0038
```

```
cat("F-statistic (from lm):", round(model_f, 4), "\n")
```

```
F-statistic (from lm): 224.0038
```

```
cat("Degrees of freedom:", df_reg, "and", df_res, "\n")
```

```
Degrees of freedom: 1 and 498
```

The F-statistic is *large and significant*, indicating that the model fits significantly better than the null model (just using $\bar{Y}$).

### 4.2.5 A Shiny App

This section provides an interactive Shiny app for exploring properties of the OLS estimator.

- Use the sidebar controls to adjust the data-generating process and observe how the sample regression function (SRF) relates to the population regression function (PRF).
- The Monte Carlo tab lets you simulate repeated sampling to see the distribution of OLS estimates.

**What to Look For When Varying Parameters**

**Increasing the error standard deviation ($\sigma_\epsilon$):**

- The scatter around the PRF increases — points fan out from the true line.
- The SRF deviates more from the PRF; residuals grow.
- $R^2$ decreases — the model explains less of the variation in $Y$.
- In the Monte Carlo tab, the sampling distribution of $b$ becomes wider. The estimator is still *unbiased* ($E(b) = \beta$), but individual estimates are less precise. This is exactly what $var(b) = \sigma^2 / \sum(X_i - \bar{X})^2$ predicts.

**Increasing the standard deviation of $X$ ($\sigma_X$):**

- The data spread out along the $X$-axis, giving the regression more "leverage."
- $R^2$ increases — more variation in $X$ means the systematic component $\beta X$ explains a larger share of total variation.
- The Monte Carlo sampling distribution of $b$ becomes *narrower*. More spread in $X$ increases $\sum(X_i - \bar{X})^2$, which shrinks $var(b)$. This is the efficiency gain from greater variation in the independent variable.

**Increasing the sample size ($n$):**

- The SRF converges toward the PRF — with more data, we estimate $\alpha$ and $\beta$ more accurately.
- The Monte Carlo distribution of $b$ tightens around $\beta$. Standard errors shrink roughly proportional to $1/\sqrt{n}$.
- $R^2$ stabilizes closer to its population value.

**Changing $\beta$ (the slope):**

- A larger $|\beta|$ steepens the PRF and increases the signal relative to the noise.
- $R^2$ increases because the systematic component of $Y$ becomes larger relative to the error.
- Setting $\beta = 0$ shows what happens under the null hypothesis — the SRF still estimates *some* slope due to sampling error, but the Monte Carlo distribution centers on zero.

# Chapter 5

# Multivariate Regression and Linear Algebra

This brief overview introduces you to vectors and matrices, components necessary in *linear algrebra*. While the linear regression model can be written in either scalar or matrix forms (or both), *linear algebra* rather than *scalar algebra*, will generally make our lives easier, though there are some elementary operations and functions we'll need to use. This chapter provides an initial overview, which we'll expand upon in later lectures.

Please note these lecture notes closely follow the notation, formulae, expressions, etc in

**Gill, Jeff. 2006. *Essential Mathematics for Political and Social Research.* Cambridge University Press.**

In particular, this chapter closely tracks chapters 3 and 4.. Please consult this source for more information. As a secondary source, please consult:

**Moore, Will and David Siegel. 2013. *A Mathematics Course for Political and Social Research.* Princeton, NJ: Princeton University Press.**

## 5.1 Introduction

Much of quantitative political science—and the social sciences more generally—aims to quantify the relationships between multiple variables. For instance, say we are interested in predicting the probability of voting, $pr(Vote)$, given one's party identification ($X_{PID}$) and political ideology ($X_{Ideology}$). Let's assume we observe these variables in the form of a survey, in which people report whether they voted, their party identification (on a seven-point scale from 1, Strong

Democrat, to 7, Strong Republican), and their ideology on a seven-point scale from 1 (Strong Liberal) to 7 (Strong Conservative). There are many ways we might address this question. For instance, we might count the number of people who vote who are Republican, the number who are conservative, and so forth. From this, we could calculate statistics such as the proportion of Democrats who vote.

Regardless of how we approach this issue, we must first understand the data itself. The data, expressed in a n excel spreadsheet, csv, or other data form, can be envisioned in matrix form. Each row of the data corresponds to a respondent (so if there are $n = 1000$ respondents, there are 1000 rows). The data is *tabular*. Each column corresponds to an observed variable—here, ideology, PID, and voting.

$$\begin{bmatrix} Vote & PID & Ideology \\ \hline a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} \end{bmatrix}$$

In this case, we have an $n \times 3$ matrix. We could generate a simple linear expression: $y_{voting} = b_0 + b_1 x_{Ideology} + b_2 x_{PID}$. Let's consider what these components entail.

The problem is that there are three unknown quantities we need to find: $b_0, b_1, b_2$.

$$\begin{bmatrix} y_1 = & b_0 + b_1 x_{Ideology,1} + b_2 x_{PID,1} \\ y_2 = & b_0 + b_1 x_{Ideology,2} + b_2 x_{PID,2} \\ y_3 = & b_0 + b_1 x_{Ideology,3} + b_2 x_{PID,3} \\ y_4 = & b_0 + b_1 x_{Ideology,4} + b_2 x_{PID,4} \\ \vdots & \\ y_n = & b_0 + b_1 x_{Ideology,n} + b_2 x_{PID,n} \end{bmatrix}$$

We have $n$ equations—one for each observation—but fewer unknowns. We need to develop tools to solve this system of equations. In other words, what constitutes a reasonably good guess of $b_0, b_1, b_2$? There are infinitely many potential values, but we'd like the one that most accurately predicts $y$. After all, we want a reasonable prediction of $y$ given our covariates—ideology and party ID. Put another way, we expect our prediction of $y$ knowing these covariates is more accurate than our prediction of $y$ absent these covariates.

This is just one reason why we need to explore the properties of matrices. Over the next couple of weeks, we'll explore linear algebra and develop techniques to solve problems like the one above. Some of this may seem obscure; like much of mathematics, it may not be immediately clear why we're exploring these techniques. Some of you may convince yourselves this is tangential to what you'll need in this program. This is incorrect—as an instructor of two of the four required methods courses, I promise every aspect of what we'll examine will

resurface. That said, don't become overly frustrated. Some material may be unfamiliar, and maybe it won't all make sense immediately. That's okay and expected. With practice, I think you'll find this material isn't all that technical. Often these concepts are best understood through applications and practice.

Typically, when we represent any matrix, the first number corresponds to the row, the second to the column. If we called this matrix $\mathbf{A}$, we could use the notation $\mathbf{A}_{N\times 3}$. More generically, we might represent an $\mathbf{A}_{N\times N}$ matrix as:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

It is worthwhile to consider what constitutes the entries of this matrix. We can think of a matrix as being made up of a series of vectors. A **vector** encodes pieces of information using a string of numbers. Returning to our running example, there are three column vectors corresponding to the variables. There are 1000 row vectors, which map each individual's vote propensity, PID in a three-dimensional space. The **length** of a vector is simply how many elements it encodes. For instance, each row vector is of length 3. Another way to write this is $\mathbf{a} \in \mathbb{R}^3$, which denotes that the vector is three-dimensional.

## 5.2 Vectors

Single numbers are called scalars; vectors are different. Vectors include multiple elements. More formally, a scalar provides only a single piece of information: magnitude or strength. Vectors exist in $\mathbb{R}^k$ and encode **strength** (its norm) as well as **direction** or location. Think of plotting a straight line in a two-dimensional space, starting at the origin. The vector encodes the magnitude—how long is the line—and the direction (where is the line going). A vector is made up of multiple elements and is most easily understood geometrically.

Assume $\mathbf{a} = [3, 2, 1]$ and $\mathbf{b} = [1, 1, 1]$. Or more simply, let's work in $\mathbb{R}^2$. Consider two vectors: $\mathbf{a} = [9, 2]$ and $\mathbf{b} = [1, 1]$. These are just two locations in a two-dimensional space. The vectors traverse different paths and have different magnitudes (one is longer). A natural question is: what is the distance between these vectors? If $\mathbf{a} = [x_1, y_1]$ and $\mathbf{b} = [x_2, y_2]$, then the Euclidean distance between these vectors is:

$$\text{Distance}(a, b)^{\text{Euclidean}} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

This is really just an extension of what you likely learned in high school using the Pythagorean Theorem. It's useful to envision things geometrically rather than simply trying to remember formulas. Drop a perpendicular from (9,2) to

form a right triangle. The line connecting **a** and **b** is the hypotenuse, solved by $c^2 = a^2 + b^2$. Plug in the differences with respect to $x$ and $y$ to find the hypotenuse.

### 5.2.1 The Norm of a Vector

This is related to another important characteristic: the **norm** of a vector. This measures the distance from the beginning point to the ending point of a vector. Think of it as the length of a line starting at the origin—it's most useful to think of a vector as starting at (0,0). The norm is a measure of strength or magnitude. Envision the point defined by the vector and the distance of that point from the origin. This is really just the distance formula if we knew two vectors: the one we're interested in—call it **a**—and the zero vector, defined by (0,0). We define the norm as:

$$\|\mathbf{a}\| = \sqrt{x_1^2 + y_1^2}$$

If we know the $x, y$ coordinates for the beginning and end point of a vector, we can move it anywhere and the norm remains the same—it doesn't change length, only relative location. It should be intuitive: the length doesn't change, only its general location in space. Also:

$$\text{Distance}(a, b)^{\text{Euclidean}} = \sqrt{\|\mathbf{a}\|^2 + \|\mathbf{b}\|^2}$$

This makes intuitive sense. We can find distance using two norms, since the norm is just the vector beginning at the origin.

### 5.2.2 Vector Addition and Subtraction

From this, we can generate further operations with a relatively clear geometric basis. Vector addition and subtraction simply involve adding (or subtracting) each element:

$$\mathbf{a} - \mathbf{b} = [3 - 1, 2 - 1, 1 - 1] = [2, 1, 0]$$

or

$$\mathbf{a} + \mathbf{b} = [3 + 1, 2 + 1, 1 + 1] = [4, 3, 2]$$

In this example, the vectors are said to be **conformable** because they have the same number of elements. **We cannot subtract or add vectors of different lengths**; for example, if $\mathbf{a} = [3, 2]$ and $\mathbf{b} = [1, 1, 1]$, the vectors are non-conformable. It's useful to remember that order generally doesn't matter for addition, but it will for subtraction.

### 5.2.3 Operators and Properties

**Commutative:** $\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a}$

**Associative:** $(\mathbf{a} + \mathbf{b}) + \mathbf{c} = \mathbf{a} + (\mathbf{b} + \mathbf{c})$

The order of vector addition doesn't matter.

**Distributive:**

(1) $c(\mathbf{a} + \mathbf{b}) = c\mathbf{a} + c\mathbf{b}$

(2) $(c + d)\mathbf{a} = c\mathbf{a} + d\mathbf{a}$

**Zero**:

$\mathbf{a} + 0 = \mathbf{a} = \mathbf{a} - \mathbf{a} = 0$

$0\mathbf{a} = 0$

$1\mathbf{a} = \mathbf{a}$

Geometrically, adding two vectors, $\mathbf{a} + \mathbf{b}$, completes the third side of a triangle.

### 5.2.4 Vectors, Extended

We can extend this to higher dimensions—vectors that reside in a higher-dimensional space, $\mathbb{R}^N$, have more than two components. Let's consider $\mathbb{R}^3$: vectors with three components. If we were to plot this, we would have $x$, $y$, and $z$ axes. For two vectors $\mathbf{a}, \mathbf{b}$ with three components:

$$\text{Distance}(a, b)^{\text{Euclidean}} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

This is the Euclidean distance in $\mathbb{R}^3$. The norm is also a slight modification:

$$\|\mathbf{a}\| = \sqrt{x_1^2 + y_1^2 + z_1^2}$$

Also, notice that when we divide a vector by its magnitude (its norm), the norm of that new vector is 1. Because of this property, we have a **normed** (or **unit**) vector. This is often useful in applied settings because it standardizes the vector. Regardless of the vector's components, a normed vector always has unit length.

## 5.3 Similarity and Vector Products

Although it is easy to multiply a scalar with a vector, there are several ways to multiply vectors. Let's define multiplication in three ways: the **inner product**, the **cross product**, and the **outer product**.

### 5.3.1 The Inner Product

Suppose $\mathbf{a} = [3, 2, 1]$ and $\mathbf{b} = [1, 1, 1]$. To calculate the inner product (or **dot product**), we multiply each element and add: $[3 \cdot 1 + 2 \cdot 1 + 1 \cdot 1] = 6$. The shorthand notation is $\sum_i a_i b_i$, which reads: "multiply the $i$-th element in $\mathbf{a}$ with the $i$-th element in $\mathbf{b}$ and then sum from 1 to $k$, where $k$ is the length of the vectors" (Gill 2006, p. 87).

**The inner product is a measure of covariance—how two vectors (or variables) go together.** The correlation is the standardized covariance: it is the mean-centered inner product divided by the product of the norms of mean-centered $x$ and mean-centered $y$.

$$\text{cov}(x, y) = E[(x - \bar{x})(y - \bar{y})] = \frac{\sum(x - \bar{x})(y - \bar{y})}{n - 1} = \frac{\text{inner.product}(x - \bar{x}, y - \bar{y})}{n - 1}$$

$$r_{x,y} = \frac{\text{cov}(x, y)}{sd(x)sd(y)} = \frac{\text{inner.product}(x - \bar{x}, y - \bar{y})}{\|\mathbf{x} - \bar{\mathbf{x}}\|\|\mathbf{y} - \bar{\mathbf{y}}\|}$$

Geometrically, if two vectors are **orthogonal** (independent), the inner product is zero, denoted $\mathbf{a} \perp \mathbf{b}$. In other words, the covariance/correlation will be zero. We can see this using the law of cosines.

### 5.3.2 Covariance and Correlation

#### 5.3.2.1 The Law of Cosines

If you know two sides of a triangle and the angle where these two sides meet, you can calculate the third side using: $c^2 = a^2 + b^2 - 2ab\cos(\theta)$, where $\theta$ must be between 0 and $\pi$. Recall the circumference of a circle is $2\pi r$. Assuming a unit circle where $r = 1$, a triangle can only be formed if $\theta < 180°$, which in radians is $\pi$. We can always find the unknown side of the triangle by knowing the angle where $a$ and $b$ meet.

Now, how does this relate to the inner product? If $\mathbf{a}$ and $\mathbf{b}$ are vectors, the inner product is:

$$\text{inner.product}(\mathbf{a}, \mathbf{b}) = \|\mathbf{a}\|\|\mathbf{b}\|\cos(\theta)$$

The inner product of $\mathbf{a}$ and $\mathbf{b}$ is a function of the vector norms and the angle between them. The only way the inner product of two non-zero vectors equals zero is if $\cos(\theta) = 0$, which occurs when $\theta = \pi/2$ (90 degrees). Thus, we only have two independent or orthogonal vectors when the angle between them is 90 degrees, or $\pi/2$ radians.

Similarly:

$$\|\mathbf{a}-\mathbf{b}\|^2 = \|\mathbf{a}\|^2 + \|\mathbf{b}\|^2 - 2\cos(\theta)$$

Rearranging:

$$\cos(\theta) = \frac{\text{inner.product}(\mathbf{a},\mathbf{b})}{\|\mathbf{a}\|\|\mathbf{b}\|}$$

$$\theta = \arccos\left(\frac{\text{inner.product}(\mathbf{a},\mathbf{b})}{\|\mathbf{a}\|\|\mathbf{b}\|}\right)$$

**The inner product is generally viewed as a measure of association.** Note the relationship between the inner product and covariance between two variables. If that covariance is zero, geometrically it means a right angle is formed.

#### 5.3.2.2 A Practical Digression: Cohesion and Voting

Why should you care? The vector norm is a measure of magnitude, strength, consistency, length, and so on. We could view it as an indicator of cohesion. Gill (2006) cites Casteveans (1970). Suppose we have a legislative vote where Conservatives vote 2, 100 and Liberals vote 72, 33. We could calculate the normed values, which gives a measure of cohesion. We could also calculate $\theta$ and compare it to other votes.

#### 5.3.2.3 Inner Product Rules

The **inner product rules**(Gill 2006)

**Commutative:** $\mathbf{a}\cdot\mathbf{b} = \mathbf{b}\cdot\mathbf{a}$

**Associative:** $d(\mathbf{a}+\mathbf{b}) = (d\mathbf{a})\cdot\mathbf{b} = \mathbf{a}\cdot d(\mathbf{b})$

**Distributive:** $\mathbf{c}(\mathbf{a}+\mathbf{b}) = \mathbf{ca}+\mathbf{cb}$

**Zero:** $\mathbf{a}\cdot 0 = 0$

**Unit:** $1\cdot\mathbf{a} = \sum a_i$

### 5.3.3 Looking Forward: The Cross Product

The cross product is another common matrix multiplication operation. Follow these steps:

1. Stack the vectors.
2. Pick a row or column.
3. Calculate the determinant on the smaller $2\times 2$ submatrix by deleting the $i$-th entry.

Say we have $\mathbf{a} = [3, 2, 1]$ and $\mathbf{b} = [1, 4, 7]$:

$$\begin{bmatrix} 3 & 2 & 1 \\ 1 & 4 & 7 \end{bmatrix}$$

The cross product is $[2 \cdot 7 - 4 \cdot 1, 1 \cdot 1 - 3 \cdot 7, 3 \cdot 4 - 2 \cdot 1]$. The important indexing trick is that if the entry is in an even-numbered location, we flip the order of subtraction. So we get $[10, -20, 10]$. The cross product is orthogonal to both original vectors (Gill 2006). We'll see that the cross-product is useful for calculating the inverse of a matrix through the determinant.

### 5.3.4 Looking Forward: The Outer Product

The outer product involves taking one vector, transposing it, and then multiplying that vector by the second vector:

$$\begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 4 & 7 \end{bmatrix}$$

If we multiply these by taking the $i$-th row and multiplying by the $j$-th column, then adding, we get:

$$\begin{bmatrix} 3 & 12 & 21 \\ 2 & 8 & 14 \\ 1 & 4 & 7 \end{bmatrix}$$

## 5.4 Matrices

As noted, we can combine row or column vectors in a matrix. For an $i \times j$ matrix, the first number typically corresponds to the number of rows, the second to the number of columns. Let's follow the convention that vectors are written as lowercase bold letters and matrices as uppercase bold letters. Like vectors, it's important to establish different types of matrices, their properties, and matrix operations.

### 5.4.1 Matrix Properties and Types

**The Equality Property:** Two matrices are equal, $\mathbf{A} = \mathbf{B}$, if all elements are identical.

**A Square Matrix:** A square matrix has an equal number of rows and columns.

**A Symmetric Matrix:** A symmetriic the same entries above and below the diagonal.

**A Skew-symmetric Matrix:** A symmetric matrix where the signs of the off-diagonals change (entries are the same, but signs differ).

**Transpose:** The transpose of a matrix changes rows to columns, or vice versa. This is denoted $\mathbf{A}^T$ or $\mathbf{A}'$.

**Squaring:** Squaring a matrix is the same as multiplying it by itself: $\mathbf{A}^2 = \mathbf{AA}$. The mechanics differ from multiplying two scalars.

**Idempotent Matrix:** $\mathbf{A}^2 = \mathbf{AA} = \mathbf{A}$

**Identity Matrix:** $\mathbf{I}$ is analogous to multiplying a scalar by 1. So $\mathbf{AI} = \mathbf{A}$. The identity matrix has zeros off the diagonal and 1s on the diagonal.

**Trace:** The trace of a matrix is the sum of all diagonal elements: $\text{tr}(\mathbf{I})$ equals the number of rows in the matrix.

### 5.4.2 Matrix Addition and Subtraction

**Always examine the matrices used in your analysis. What are the number of rows? What are the number of columns? What data are missing?**

How do we perform operations on matrices? After all a **matrix** encodes far more than a **scalar** and is a combination of *row* or *column* vectors.

For matrices to be added or subtracted, they must be **conformable** (same dimensions). To add or subtract, simply add (or subtract) the $i, j$ elements:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \cdots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \cdots & a_{2n} + b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} + b_{n1} & a_{n2} + b_{n2} & \cdots & a_{nn} + b_{nn} \end{bmatrix}$$

A matrix multiplied by a scalar is every element multiplied by that scalar.

**Commutative:** $\mathbf{X} + \mathbf{Y} = \mathbf{Y} + \mathbf{X}$

**Additive:** $(\mathbf{X} + \mathbf{Y}) + \mathbf{Z} = \mathbf{X} + (\mathbf{Y} + \mathbf{Z})$

**Distributive (Matrix):** $\mathbf{c}(\mathbf{X} + \mathbf{Y}) = \mathbf{cX} + \mathbf{cY}$

**Distributed (Scalar):** $(c + t)\mathbf{X} = c\mathbf{X} + t\mathbf{X}$

**Zero:** $\mathbf{A} + 0 = \mathbf{A}$

### 5.4.3 Matrix Multiplication

We need to elaborate on this general algebra to multiply two matrices and invert matrices. A critical component that makes matrix multiplication different from scalar multiplication is that **order absolutely matters**. It is entirely

conceivable that **AB** differs from **BA**. It is also possible that **AB** can be multiplied but **BA** cannot!

To multiply two matrices, we multiply and add the $i$-th row with the $j$-th column:

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} 3 & 5 \\ 2 & 4 \end{bmatrix} = \begin{bmatrix} 1 \cdot 3 + 3 \cdot 2 & 1 \cdot 5 + 3 \cdot 4 \\ 2 \cdot 3 + 4 \cdot 2 & 2 \cdot 5 + 4 \cdot 4 \end{bmatrix}$$

If we multiply two $2 \times 2$ matrices, we get a $2 \times 2$ matrix. Two matrices are conformable for multiplication if the first matrix has the same number of columns as the second has rows. The dimensions of the resulting matrix are the rows of the first matrix by the columns of the second.

For example: - $2 \times 2$ multiplied by $3 \times 2$ is not conformable. - $3 \times 2$ multiplied by $2 \times 2$ yields a $3 \times 2$ matrix. - $9 \times 2$ multiplied by $2 \times 11$ yields a $9 \times 11$ matrix.

This is just one reason why order matters.

## 5.5 Matrix Conformability

**Conformability** refers to whether two matrices have compatible dimensions for a particular matrix operation. Different operations have different requirements. Knowing whether a matrix or matrices are conformable in a particular operation is essential to debuggin.

### 5.5.1 Conformability for Addition and Subtraction

Recall, for addition and subtraction, matrices must have **exactly the same dimensions**—the same number of rows and the same number of columns. If **A** is $m \times n$ and **B** is $m \times n$, then $\mathbf{A} + \mathbf{B}$ and $\mathbf{A} - \mathbf{B}$ are both $m \times n$.

**Example (Conformable):**

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}_{2\times2} \quad \mathbf{B} = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}_{2\times2}$$

$$\mathbf{A} + \mathbf{B} = \begin{bmatrix} 1+5 & 2+6 \\ 3+7 & 4+8 \end{bmatrix} = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}_{2\times2}$$

**Example (Non-conformable):**

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}_{2\times2} \quad \mathbf{C} = \begin{bmatrix} 5 & 6 & 7 \\ 8 & 9 & 10 \end{bmatrix}_{2\times3}$$

$\mathbf{A} + \mathbf{C}$ is not defined—dimensions don't match

### 5.5.2 Conformability for Multiplication

For matrix multiplication, the rule is: **the number of columns in the first matrix must equal the number of rows in the second matrix.**

If $\mathbf{A}$ is $m \times n$ and $\mathbf{B}$ is $n \times p$, then $\mathbf{AB}$ is conformable and produces an $m \times p$ matrix.

**General Rule:**

$$\mathbf{A}_{m\times n} \times \mathbf{B}_{n\times p} = \mathbf{C}_{m\times p}$$

The inner dimensions ($n$ in both matrices) must match. The result has dimensions of the outer dimensions ($m$ rows from $\mathbf{A}$, $p$ columns from $\mathbf{B}$).

**Example (Conformable):**

$$\mathbf{A}_{2\times 3} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad \mathbf{B}_{3\times 2} = \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix}$$

The first matrix has 3 columns, the second has 3 rows—**conformable for multiplication**.

$$\mathbf{AB}_{2\times 2} = \begin{bmatrix} 1(7)+2(9)+3(11) & 1(8)+2(10)+3(12) \\ 4(7)+5(9)+6(11) & 4(8)+5(10)+6(12) \end{bmatrix}$$

$$= \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix}_{2\times 2}$$

**Example (Non-conformable):**

$$\mathbf{A}_{2\times 3} \times \mathbf{C}_{2\times 2}$$

The first matrix has 3 columns, but the second has only 2 rows—**not conformable**. You cannot multiply these matrices.

### 5.5.3 Order Matters

Conformability reveals why matrix multiplication is **not commutative** – order matters.

**Case 1:** $\mathbf{A}_{2\times 3}$ and $\mathbf{B}_{3\times 2}$

- **AB**: Columns of **A** (3) = Rows of **B** (3) **Conformable** → Result is $2 \times 2$
- **BA**: Columns of **B** (2) Rows of **A** (2)… wait, that's equal! **Conformable** → Result is $3 \times 3$

But **AB** and **BA** produce **different dimensions and different values**.

**Case 2:** $\mathbf{A}_{2\times3}$ and $\mathbf{C}_{2\times2}$

- **AC**: Columns of **A** (3) Rows of **C** (2) **Not conformable**
- **CA**: Columns of **C** (2) Rows of **A** (2)… wait, that's equal! **Conformable** → Result is $2 \times 3$

## 5.6 The Transpose

The transpose of a matrix $\mathbf{A}$, written $\mathbf{A}^T$ (or $\mathbf{A}'$), swaps rows and columns. If $\mathbf{A}$ is $m \times n$, then $\mathbf{A}^T$ is $n \times m$.

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}_{2\times3} \quad \Longrightarrow \quad \mathbf{A}^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}_{3\times2}$$

### 5.6.1 Key Properties of the Transpose

| Property | Statement |
|---|---|
| Double transpose | $(\mathbf{A}^T)^T = \mathbf{A}$ |
| Sum | $(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T$ |
| Scalar | $(c\mathbf{A})^T = c\mathbf{A}^T$ |
| Product (reversal) | $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$ |
| Symmetric matrix | $\mathbf{A} = \mathbf{A}^T$ iff $\mathbf{A}$ is symmetric |

The product rule is especially important: **transposing a product reverses the order**. This generalizes: $(\mathbf{ABC})^T = \mathbf{C}^T\mathbf{B}^T\mathbf{A}^T$. You'll see this repeatedly in the OLS derivation.

A useful result: for any matrix $\mathbf{A}$, the product $\mathbf{A}^T\mathbf{A}$ is always a symmetric, square matrix (with dimensions $n \times n$ if $\mathbf{A}$ is $m \times n$). This is exactly what $\mathbf{X}^T\mathbf{X}$ produces in the normal equations.

## 5.7 Matrix Inversion

For scalars, the inverse of $a$ is $a^{-1} = 1/a$, and $a \cdot a^{-1} = 1$. The matrix analog is: $\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

Only **square** matrices can have inverses, and not all square matrices do. A matrix that has an inverse is called **nonsingular** (or invertible); one that does not is **singular**.

### 5.7.1 When Does the Inverse Exist?

A square matrix $\mathbf{A}$ is invertible if and only if its **determinant** is nonzero: $\det(\mathbf{A}) \neq 0$.

For a $2 \times 2$ matrix:

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \Longrightarrow \quad \mathbf{A}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

The denominator $ad - bc$ is the determinant. If it equals zero, the matrix is singular and no inverse exists.

**Example:**

$$\mathbf{A} = \begin{bmatrix} 4 & 7 \\ 2 & 6 \end{bmatrix} \quad \Longrightarrow \quad \det(\mathbf{A}) = 4(6) - 7(2) = 10$$

$$\mathbf{A}^{-1} = \frac{1}{10} \begin{bmatrix} 6 & -7 \\ -2 & 4 \end{bmatrix} = \begin{bmatrix} 0.6 & -0.7 \\ -0.2 & 0.4 \end{bmatrix}$$

Verify: $\mathbf{A}\mathbf{A}^{-1} = \begin{bmatrix} 4(0.6) + 7(-0.2) & 4(-0.7) + 7(0.4) \\ 2(0.6) + 6(-0.2) & 2(-0.7) + 6(0.4) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{I}$

### 5.7.2 Key Properties of the Inverse

| Property | Statement |
|---|---|
| Product (reversal) | $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$ |
| Transpose | $(\mathbf{A}^T)^{-1} = (\mathbf{A}^{-1})^T$ |
| Double inverse | $(\mathbf{A}^{-1})^{-1} = \mathbf{A}$ |
| Identity | $\mathbf{I}^{-1} = \mathbf{I}$ |

Like the transpose, **inverting a product reverses the order**.

### 5.7.3 Why This Matters for OLS

In the OLS derivation, we arrive at the **normal equations**: $\mathbf{X}^T\mathbf{X}\mathbf{b} = \mathbf{X}^T\mathbf{y}$. To isolate $\mathbf{b}$, we multiply both sides by $(\mathbf{X}^T\mathbf{X})^{-1}$:

$$\mathbf{b} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

This requires $\mathbf{X}^T\mathbf{X}$ to be invertible. It fails when $\mathbf{X}$ has **perfect multicollinearity** — one column is a linear combination of others — because $\det(\mathbf{X}^T\mathbf{X}) = 0$. This is the matrix algebra reason behind Assumption 9 (no perfect multicollinearity).

## 5.8 Linear Regression and Matrix Algebra

Let's take it a step further – since at this point it may not be clear even why we need the inverse. Oftentimes in the social sciences we're interested in solving a system of equations.

$$\begin{bmatrix} y_1 = & b_0 + b_1 x_{Ideology,1} + b_2 x_{PID,1} \\ y_2 = & b_0 + b_1 x_{Ideology,2} + b_2 x_{PID,2} \\ y_3 = & b_0 + b_1 x_{Ideology,3} + b_2 x_{PID,3} \\ y_4 = & b_0 + b_1 x_{Ideology,4} + b_2 x_{PID,4} \\ \vdots & \\ y_n = & b_0 + b_1 x_{Ideology,n} + b_2 x_{PID,n} \end{bmatrix}$$

The linear regression model can be written in matrix form as:

$$\mathbf{y} = \mathbf{Xb} + \mathbf{e}$$

The dependent variable, $Y$ is a vector.

$$\mathbf{y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}$$

The matrix $X$ consists of the independent variables. The first column is a vector of ones for the intercept term, and the remaining columns are the independent variables – i.e., your data.

$$\mathbf{X} = \begin{bmatrix} 1 & X_{11} & X_{12} & \cdots & X_{1k} \\ 1 & X_{21} & X_{22} & \cdots & X_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & X_{n1} & X_{n2} & \cdots & X_{nk} \end{bmatrix}$$

The parameter vector is what we are estimating, the slopes and intercept.

$$\mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_k \end{bmatrix}$$

And the errors

$$\mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix}$$

And all together:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{12} & x_{13} & x_{14} & \cdots & x_{1k} \\ 1 & x_{22} & x_{23} & x_{24} & \cdots & x_{2k} \\ 1 & x_{32} & x_{33} & x_{34} & \cdots & x_{3k} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ 1 & x_{n2} & x_{n3} & x_{n4} & \cdots & x_{nk} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_k \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ \vdots \\ e_n \end{bmatrix}$$

And,

$$\begin{aligned} y_1 &= b_0 + b_1 x_{11} + b_2 x_{12} + \cdots + b_k x_{1k} + e_1 \\ y_2 &= b_0 + b_1 x_{21} + b_2 x_{22} + \cdots + b_k x_{2k} + e_2 \\ &\vdots \\ y_n &= b_0 + b_1 x_{n1} + b_2 x_{n2} + \cdots + b_k x_{nk} + e_n \end{aligned}$$

Which is just:

$$y_i = b_0 + b_1 x_{i1} + b_2 x_{i2} + \cdots + b_k x_{ik} + e_1$$

## 5.9 Deriving the OLS Estimator in Matrix Form

The goal is the same, minimize this:

$$min(\mathbf{e}^T\mathbf{e})$$

Where

$$\mathbf{e^T e} = (\mathbf{y} - \mathbf{Xb})^T(\mathbf{y} - \mathbf{Xb})$$

$$\mathbf{e}^T\mathbf{e} = \mathbf{y}^T\mathbf{y} - \mathbf{b}^T\mathbf{X}^T\mathbf{y} - \mathbf{y}^T\mathbf{Xb} + \mathbf{b}^T\mathbf{X}^T\mathbf{Xb}$$

But $\mathbf{b}^T\mathbf{X}^T\mathbf{y}$ and $\mathbf{y}^T\mathbf{Xb}$ are scalars, and

$$[\mathbf{y}^T\mathbf{Xb}]^T = \mathbf{b}^T\mathbf{X}^T\mathbf{y}$$

$$\mathbf{e}^T\mathbf{e} = \mathbf{y}^T\mathbf{y} - 2\mathbf{b}^T\mathbf{X}^T\mathbf{y} + \mathbf{b}^T\mathbf{X}^T\mathbf{Xb}$$

$$\frac{\partial \mathbf{e}^T\mathbf{e}}{\partial \mathbf{b}} = 0 - 2\mathbf{X^T y} + 2\mathbf{X^T Xb}$$

Again, set this to 0, and $\mathbf{X^T y} = \mathbf{X^T X b}$, the normal equations!

When we multiply a matrix by its inverse, we get the identity matrix, so multiply the equation by $(\mathbf{X^T X})^{-1}$

$$(\mathbf{X^T X})^{-1}\mathbf{X^T y} = \mathbf{b}$$

As long as $\mathbf{X^T X}$ is invertible, we can solve for vector $\mathbf{b}$.

Because the matrix version has a more compact representation, it is often easier to manage with multiple variables. The equation is always the same. It's also common to see the linear equation written in matrix form

$$\mathbf{y} = \mathbf{X b} + \mathbf{e}$$

# Chapter 6

# Model Fit, Uncertainty, and Inference

## 6.1 Model Fit and Prediction

This chapter focuses on inference. What can we learn about the the population regression function from the sample regression function? How do we make inferences about the population parameters?

The dashboard below includes several features: - The left-hand panel allows you to adjust the parameters of the data generating process (DGP) – the intercept, slope, error variance, and sample size.

- If you choose to run `monte carlo simulations` the dashboard will simulate datasets from the specified DGP, estimate the OLS coefficients, and then display the distributions.
- The dashboard `tabs` allow you to compare the OLS estimates to the true population parameters, and to examine how the OLS estimates vary across different parameter specifications.

Let's consider the OLS model by varying different parameters to see how this influences things like $R^2$

### 6.1.1 OLS Shiny App

## 6.2 From Bivariate to Multiple Regression

The OLS estimator provides a point estimate – a single "best" prediction about $y$ given $x$. We predict a single point – $\widehat{y}_i$ – given a predictor value $x_i$. With *multiple predictors*, the Sample Regression Function (SRF) expands from a line

to a plane. The error term $e_i$ is still the vertical distance from the observed $Y_i$ to the predicted $\hat{Y}_i$, but now $\hat{Y}_i$ is a linear combination of multiple predictors:

$$Y_i - \underbrace{(a + b_1 X_1 + b_2 X_2)}_{\hat{Y}_i} = e_i$$

We still find the coefficients that minimize $\sum e^2$, but now the logic changes somewhat. Instead of fitting a *line* through a scatter of points in two dimensions, we are fitting a *plane* through a cloud of scatter points. Each coefficient $b_j$ represents the expected change in $Y$ for a one-unit change in $X_j$, **holding all other predictors constant**. This "holding constant" interpretation is important for multiple regression – it allows us to isolate the partial effect of each variable. That is also we we say these are "control" variables. We are isolating the effect of one variable, while simultaneously controlling for the influence of other variables.

Figure 6.1 illustrates this with two independent variables. The black points are the observed data, the blue plane represents the fitted regression, and the red lines show the residuals – the vertical distance from each point to the plane.

The OLS coefficients are chosen to minimize the sum of squared lengths of these red lines. We could move the plane up or down, or tilt it, but there is only one solution – the minimum value.

Let's just capture this with toy data.

```
n <- 500
x1 <- rnorm(n, 3, 2)
x2 <- rnorm(n, 1, 2)
y <- 1 + 0.75 * x1 - 0.50 * x2 +
  rnorm(n, 0, 1.5)

dat <- data.frame(x1, x2, y)
fit <- lm(y ~ x1 + x2, data = dat)
dat$yhat <- fitted(fit)
```

### 6.2.1 Deriving the Multiple Regression Coefficients

We've now used two methods to derive the OLS coefficients for a bivariate regression: the covariance method and the deviation form. The same methods apply to multiple regression, but the algebra is more complex.

At the beginning of the term, we considered two variables and used scalar algebra to derive the OLS coefficients. Now, with multiple predictors, we can use matrix algebra to derive the coefficients in a more compact form; then we moved to $k$ independent variables, and the matrix solution is the most efficient way to derive the OLS coefficients.

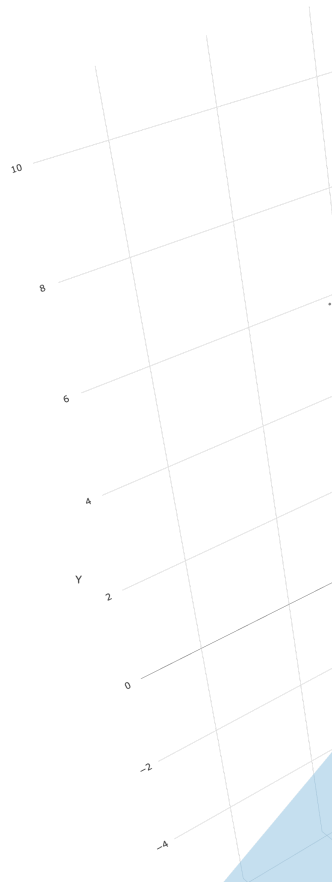



Figure 6.1: The OLS regression plane in three dimensions. Black points are observations and the shaded surface is the fitted plane.

Recall that in deviation form, where $y_i = Y_i - \bar{Y}$, $x_{1i} = X_{1i} - \bar{X}_1$, and $x_{2i} = X_{2i} - \bar{X}_2$, we set the partial derivatives of the sum of squared residuals to zero. Setting $\frac{\partial SSR}{\partial b_1} = 0$:

$$\begin{aligned} 0 &= -2\sum(Y_i - a - b_1 X_{1i} - b_2 X_{2i})(X_{1i}) \\ b_1 &= \frac{\sum x_{1i} y_i - b_2 \sum x_{1i} x_{2i}}{\sum x_{1i}^2} \end{aligned}$$

Setting $\frac{\partial SSR}{\partial b_2} = 0$:

$$\begin{aligned} 0 &= -2\sum(Y_i - a - b_1 X_{1i} - b_2 X_{2i})(X_{2i}) \\ b_2 &= \frac{\sum x_{2i} y_i - b_1 \sum x_{1i} x_{2i}}{\sum x_{2i}^2} \end{aligned}$$

Solving simultaneously yields the closed-form solutions:

$$\begin{aligned} b_1 &= \frac{\sum y_i x_{1i} \sum x_{2i}^2 - \sum x_{1i} x_{2i} \sum x_{2i} y_i}{\sum x_{1i}^2 \sum x_{2i}^2 - (\sum x_{1i} x_{2i})^2} \\ b_2 &= \frac{\sum y_i x_{2i} \sum x_{1i}^2 - \sum x_{1i} x_{2i} \sum x_{1i} y_i}{\sum x_{1i}^2 \sum x_{2i}^2 - (\sum x_{1i} x_{2i})^2} \end{aligned}$$

Notice the denominator is the same for both coefficients. **It equals zero when $X_1$ and $X_2$ are perfectly collinear** – in which case the system has no unique solution. This foreshadows the multicollinearity problem.

Because we've written everything in deviation form, let's just write this in terms of covariances and variances:

$$\begin{aligned} b_1 &= \frac{cov(Y, X_1) var(X_2) - cov(X_1, X_2) cov(Y, X_2)}{var(X_1) var(X_2) - cov(X_1, X_2)^2} \\ b_2 &= \frac{cov(Y, X_2) var(X_1) - cov(X_1, X_2) cov(Y, X_1)}{var(X_1) var(X_2) - cov(X_1, X_2)^2} \end{aligned}$$

So notice what the equation provides. $b_1$, for instance, is based on several things

1. The covariance between $Y$ and $X_1$ (the bivariate relationship)
2. The variance of $X_2$ (the "control" variable)
3. The covariance between $X_1$ and $X_2$ (the correlation between the predictor of interest and the control variable)

If $X_1$ and $X_2$ are uncorrelated, the second term in the numerator drops out, and the denominator simplifies to $var(X_1)var(X_2)$, so $b_1$ reduces to

$$b_1 = \frac{cov(Y, X_1)}{var(X_1)}$$

which is the bivariate slope.

But if $X_1$ and $X_2$ are correlated, the second term in the numerator adjusts for that correlation, and the denominator accounts for the shared variance between $X_1$ and $X_2$.

> **!** **The Importance of Model Specification**
>
> In order to accurately estimate the partial effect of $X_1$ on $Y$, we need to include all relevant predictors that are correlated with both $X_1$ and $Y$. The Sample Regression Function (SRF) must match the Population Regression Function (PRF). This means we need to include all relevant variables that **affect** $y$ and are correlated with $X_1$. Proper specification **does not mean** including **all** variables that we think affect $y$. Oversaturating the model will:
>
> 1. Inflate standard errors, making it difficult to obtain precise estimates.
> 2. Not necessarily improve the model's ability to estimate the partial effect of $X_1$ on $Y$and can produce **overfitted** models. See below...

### 6.2.2 Linear Algebra

Of course, with $k$ predictors this all becomes needlessly tedious. So instead,

$$\mathbf{b} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

The requirement is that $\mathbf{X}^T\mathbf{X}$ be invertible, which fails under perfect multicollinearity – when one column of $\mathbf{X}$ is an exact linear combination of the others. Table 6.1 verifies that if we write it out, it's the same as the `lm()` solution.

## 6.3 Interpretation of Coefficients in Multiple Regression

The linear regression model should be estimated with a **continuous dependent variable** (recall the assumption the independent and identically distributed errors). However, the independent variable(s) may be quantitative (i.e., continuous) or qualitative (i.e.,categorical).

Table 6.1

```
# Verify: matrix solution matches lm()
X <- cbind(1, x1, x2)
b_matrix <- solve(t(X) %*% X) %*% t(X) %*% y
cbind(matrix_solution = b_matrix, lm_solution = coef(fit))
```

```
             lm_solution
    0.8487808   0.8487808
x1  0.7800368   0.7800368
x2 -0.4992539  -0.4992539
```

For a continuous independent variable, the interpretation of the coefficient is straightforward: the slope coefficient corresponds to the expected change in $Y$ for a one-unit increase in $X$, holding all other predictors constant. The slope is the same for every observation, meaning that the relationship between $X$ and $Y$ are said to be **linear and additive**. This phrase will come up often in POL 683, but for now, it simply means that the relationship between $x$ and $y$ does not change across the range of $x$ or any other variables in the regression model.

In other words, it means

$$dy/dx = b$$

for all values of $x$ and all values of the other predictors. The effect of $X$ on $Y$ is constant across the data.

### 6.3.1 In Practical Terms

**Linearity and Additivity** means that it is relatively simple to glean quite a lot of information from a linear regression table (see Table 6.3).

**Data: 2020 Western States Survey**

The examples throughout this chapter use the **2020 Western States Survey (WSS)**, a YouGov survey of 3,000 respondents across western U.S. states (Arizona, California, Colorado, Nevada, New Mexico, Idaho, Montana, Oregon, Utah, Washington, Wyoming), including a Latino oversample of 600 respondents. The sample was matched on gender, age, race, and education, and weighted using propensity scores and post-stratification. The survey was fielded around the 2020 presidential election.

**Institutional trust** is a composite of seven items, each measured on a 4-point scale (1 = no trust at all, 4 = a great deal of trust): trust in Congress, the President, the Supreme Court, the federal government, state legislatures, the police, and science. The composite averages across the seven items and is rescaled to 0–1.

**Authoritarianism** is the mean of four binary items (`auth_1`–`auth_4`), yielding a 0–1 scale where higher values indicate greater authoritarianism.

**Party identification** is a three-category measure: Republican, Independent, and Democrat, coded as dummy variables with Independents as the reference category.

The alpha statistic measures internal consistency – how well the items in a scale hang together. Values above 0.70 are generally considered acceptable for a composite measure.

Let's use the institutional trust scale to illustrate the interpretation of coefficients. Both the dependent variable (institutional trust) and authoritarianism vary from 0 to 1, which makes interpretation straightforward: a one-unit change corresponds to moving from the minimum to the maximum of each scale.

**It's always a good idea to examine the scale and characteristics of your variables prior to more complex analysis.**

Let's take a closer look at the party identification variables. They are coded as binary indicators (0 or 1).

## 6.4 Dummy Variables and Categorical Predictors

OLS is restrictive in the sense that the dependent variable should generally be continuous, but the independent variables can be continuous or categorical.

With a categorical predictor, we encode group membership using **dummy variables** – binary indicators coded 0 or 1. With $k$ categories, we include $k-1$ dummies; the omitted group serves as the **reference category**. Why omit one? One is is a perfect linear combination of the others.

Table 6.2

```
library(dplyr)
library(tidyr)

download.file("https://raw.githubusercontent.com/crweber9874/advancedRegression/main/da
              destfile = "wss20.rda")
load("wss20.rda")

wss20 <- wss20 |>
  pivot_wider(names_from = contestation, values_from = contestation_value) |>
  mutate(
    authoritarianism = rowMeans(cbind(auth_1, auth_2, auth_3, auth_4), na.rm = TRUE),
    democrat = ifelse(party_identification3 == 1, 1, 0),
    republican = ifelse(party_identification3 == 3, 1, 0),
    independent = ifelse(party_identification3 == 2, 1, 0)
  )

# Cronbach's alpha for the 7-item trust scale
trust_items <- wss20 |>
  select(trustCongress, trustPresident, trustSC, trustGovernment,
         trustStateleg, trustPolice, trustScience) |>
  na.omit()

k <- ncol(trust_items)
item_vars <- apply(trust_items, 2, var)
total_var <- var(rowSums(trust_items))
alpha <- (k / (k - 1)) * (1 - sum(item_vars) / total_var)
cat("Cronbach's alpha for institutional trust (7 items):", round(alpha, 3), "\n")
```

```
Cronbach's alpha for institutional trust (7 items): 0.701
```

Table 6.3

```
# Build the trust composite and rescale to 0-1
wss20 <- wss20 |>
  mutate(
    trust_raw = rowMeans(cbind(trustCongress, trustPresident, trustSC,
                               trustGovernment, trustStateleg, trustPolice,
                               trustScience), na.rm = TRUE),
    institutional_trust = (trust_raw - 1) / 3  # rescale from 1-4 to 0-1
  )

fit_trust <- lm(institutional_trust ~ survey_wave + presvote_trump_2020 + presvote_trump_2020+  a
summary(fit_trust)
```

```
Call:
lm(formula = institutional_trust ~ survey_wave + presvote_trump_2020 +
    presvote_trump_2020 + authoritarianism, data = wss20)

Residuals:
     Min       1Q   Median       3Q      Max
-0.62223 -0.10847  0.01194  0.11696  0.56091

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         0.464002   0.008094  57.329  < 2e-16 ***
survey_wavepre     -0.024910   0.007787  -3.199  0.00139 **
presvote_trump_2020 0.107560   0.007021  15.320  < 2e-16 ***
authoritarianism    0.050672   0.010055   5.040 4.95e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1755 on 2876 degrees of freedom
  (720 observations deleted due to missingness)
Multiple R-squared:  0.106, Adjusted R-squared:  0.105
F-statistic: 113.6 on 3 and 2876 DF,  p-value: < 2.2e-16
```

For instance, say party identification is coded as 1 = *Republican*, 2 = *Independent*, and 3 = *Democrat*. We shouldn't include this as an independent variable, because it doesn't make a lot of sense. *What is a unit change*? Is the movement from Republican to Independent the same as the movement from Independent to Democrat? Are Independents equidistant – in terms of ideology, voting, political behavior – from Republicans and Democrats?

Instead, let's not assume this restrictive structure and create **dummy variables** for each category. With $k$ categories, we include $k-1$ dummies; the omitted group serves as the **reference category**. Recall that in a regression model with an intercept this is the point when all the independent variables are 0. Well, an Independent according to this scheme is a case when both the Republican and Democrat dummies are 0. If we were to also include a dummy for Independents, then we would have perfect multicollinearity – the dummy variables would be perfectly linearly dependent on each other and the intercept.

### 6.4.1 Intercept Shifts

Consider a model with a continuous predictor (authoritarianism) and party identification dummies (Republican, Democrat), with Independents as the reference category:

$$Y_i = \alpha + \gamma_{\text{Rep}} D_{\text{Rep}} + \gamma_{\text{Dem}} D_{\text{Dem}} + \beta_{\text{Auth}} X_{\text{Auth}} + e_i$$

The expected values for each group are:

$$\begin{aligned} E(Y \mid \text{Independent}) &= \alpha + \beta_{\text{Auth}} X_{\text{Auth}} \\ E(Y \mid \text{Republican}) &= (\alpha + \gamma_{\text{Rep}}) + \beta_{\text{Auth}} X_{\text{Auth}} \\ E(Y \mid \text{Democrat}) &= (\alpha + \gamma_{\text{Dem}}) + \beta_{\text{Auth}} X_{\text{Auth}} \end{aligned}$$

The slope on authoritarianism is the **same** for every group – only the intercept shifts. The coefficients $\gamma_{\text{Rep}}$ and $\gamma_{\text{Dem}}$ represent the average difference in $Y$ between each group and the reference category (Independents), holding authoritarianism constant.

While the choice of the baseline is arbitrary in terms of $\widehat{Y}_i$, it is meaningful in terms of how you interpret $\gamma_1 ... \gamma_k$.

The three parallel lines in Figure 6.2 illustrate how the intercept moves about: all three groups share the same slope on authoritarianism, but they differ in their baseline levels of institutional trust.

Table 6.4

```
fit_dummy <- lm(institutional_trust ~ authoritarianism + republican + democrat, data = wss20)
summary(fit_dummy)
```

```
Call:
lm(formula = institutional_trust ~ authoritarianism + republican +
    democrat, data = wss20)

Residuals:
     Min       1Q   Median       3Q      Max
-0.60364 -0.11477  0.01496  0.12002  0.58308

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      0.403791   0.008483  47.601  < 2e-16 ***
authoritarianism 0.052506   0.009472   5.543 3.20e-08 ***
republican       0.147347   0.009239  15.949  < 2e-16 ***
democrat         0.044315   0.008715   5.085 3.88e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1836 on 3427 degrees of freedom
  (169 observations deleted due to missingness)
Multiple R-squared:  0.1051,    Adjusted R-squared:  0.1044
F-statistic: 134.2 on 3 and 3427 DF,  p-value: < 2.2e-16
```

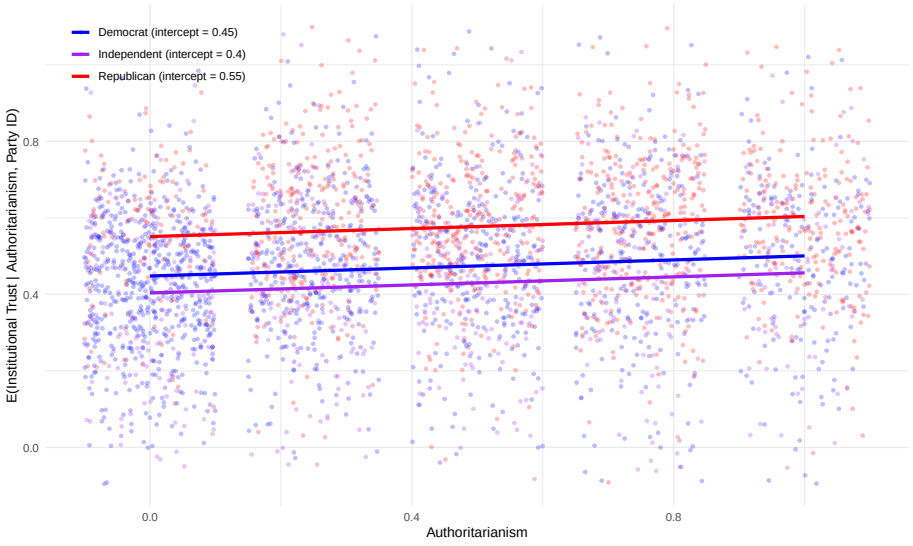



Figure 6.2: Intercept shifts by party identification. All three groups share the same slope on authoritarianism.

## 6.5 Revisiting Model Fit: $R^2$ in Multiple Regression

Recall that we established that the total sum of squares can be decomposed into the regression and residual sum of squares: $TSS = RegSS + RSS$:

$$R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{\sum(Y_i - \hat{Y}_i)^2}{\sum(Y_i - \bar{Y})^2}$$

A critical issue is that $R^2$ **never decreases** when you add a predictor – even a useless one.

### 6.5.1 Adjusted $R^2$

We can adjust for this problem by using the **adjusted** $R^2$, which penalizes for model complexity. Adding predictors that explain little variation in $y$ may lead to a reduction in this value.

$$\bar{R}^2 = 1 - \frac{RSS/(n-k-1)}{TSS/(n-1)} = 1 - (1 - R^2)\frac{n-1}{n-k-1}$$

### 6.5.2 The Overfitting Problem

**In-sample** fit ($R^2$). This is the value we obtain from our regression model. The model parameters are estimated on these values of $x$ and $y$. As we add more and more variables, $R^2$ increases. With $n-1$ predictors and $n$ observations, $R^2 = 1$. It becomes difficult then to discern whether inclusion of a variable actually improves the model's ability to explain variation in $y$, or whether it just capitalizes on randomness in data. One solution – and a relatively simple one at that – is to estimate the model paramters on one dataset, and then evaluate the model's performance on a different dataset.

**Out-of-sample** prediction measures how well the model generalizes to *new, unseen data*. What is our $R^2$ when we apply the model to a different dataset, using the parameter estimates from the original model? Is there a substantial discrepancy?

If our $R^2$ is high in-sample but low out-of-sample, this suggests that the model is overfitting – it is capturing noise in the training data rather than the underlying signal. The gap between in-sample and out-of-sample performance is a diagnostic for overfitting.

### 6.5.3 K-Fold Cross-Validation

Cross-validation estimates out-of-sample prediction error using only the available data [Hastie et al., 2009]. The procedure works as follows.

1. Randomly partition the data into $K$ roughly equal-sized groups, called **folds**
2. For each fold $k = 1, \ldots, K$: **train** the model on all data except fold $k$. So if you partitioned the data into 3 folds, you might train the data on folds 1 and 2, and then test the model on fold 3. Next, you might train the model on folds 1 and 3, and test on fold 2. Finally, you might train the model on folds 2 and 3, and test on fold 1. In each case, you are training the model on a different subset of the data, and testing it on the remaining fold.

More compactly, **predict** on fold $k$, and compute $\text{MSE}_k = \frac{1}{n_k} \sum_{i \in \text{fold } k} (Y_i - \hat{Y}_i)^2$

3. Finally, average across folds: $\text{CV}_{(K)} = \frac{1}{K} \sum_{k=1}^{K} \text{MSE}_k$

It's common to see $K = 10$. When $K = n$, this is called **leave-one-out cross-validation** (LOOCV). It progresses in the same manner, but will instead iterate through the entire dataset, leaving one test observation out at a time.

> **Why MSE and not $R^2$?**
>
> It's common to use the MSE rather than $R^2$ for cross-validation because $R^2$ is defined relative to $\bar{Y}$ and $TSS$ from the *training* data. When predicting on a held-out fold, that fold has its own $\bar{Y}$. The MSE is useful because it just tells us how much we're off on average.

Note below that we're doing a few things. First, we're defining the number of folds. Then we're sampling from these $k$ folds $n$ times to assign each unit to a fold. Next, we're proceeding through a loop which simply defines the test and training set, calculates the MSE result, saves it, then moves on to the next fold. Finally, we display everything in a data frame.

```
set.seed(42)
cv_data <- wss20 |>
    filter(!is.na(institutional_trust), !is.na(authoritarianism),
                            !is.na(republican), !is.na(democrat))

K <- 10

folds <- sample(rep(1:K, length.out = nrow(cv_data)))

mse_simple <- mse_full <- numeric(K)

for (k in 1:K) {
  train <- cv_data[folds != k, ]
  test  <- cv_data[folds == k, ]
  fit_s <- lm(institutional_trust ~ authoritarianism, data = train)
  mse_simple[k] <- mean((test$institutional_trust - predict(fit_s, test))^2)
  fit_f <- lm(institutional_trust ~ authoritarianism + republican + democrat, data = tr
  mse_full[k] <- mean((test$institutional_trust - predict(fit_f, test))^2)
}

data.frame(Model = c("Authoritarianism only", "+ Party ID dummies"),
           CV_MSE = round(c(mean(mse_simple), mean(mse_full)), 4)) |>
  knitr::kable()
```

Table 6.5: 10-fold cross-validation MSE comparison.

| Model | CV_MSE |
|---|---|
| Authoritarianism only | 0.0368 |
| + Party ID dummies | 0.0338 |

If the fuller model has lower CV-MSE (Table 6.5), it genuinely improves prediction – not just in-sample fit.

$$\text{RMSE} = \sqrt{0.6143} \approx 0.78$$

### 6.5.4 The Lewis-Beck vs. Achen Debate

Scholars disagree about the role of $R^2$ in evaluating models. Recall that**Lewis-Beck and Skalaban (1990)** argued that $R^2$ is a useful and informative measure of model fit: a high $R^2$ indicates that the model accounts for a substantial share of variance in $Y$, and comparing $R^2$ across models helps assess whether new predictors contribute. They emphasize that it serves as useful, comparable metric to compare how well a linear regression model performs.

**Achen (1982, 1990)** countered that $R^2$ is misleading and overemphasized. The argument is strongly tied to the notion of overfitting.

- $R^2$ depends on the variance of $X$ in the sample – the same causal effect can yield very different $R^2$ values in different datasets
- Researchers may observe inflated $R^2$ by choosing samples with high variance in $X$, or by adding irrelevant predictors
- $R^2$ does not measure *causal* impact and may not be useful for testing substantively important theories in our field.

The middle ground: $R^2$ is useful for **prediction** – how well does the model forecast $Y$? Adjusted $R^2$ and cross-validation are better tools for model comparison than raw $R^2$. Methods like cross-validation are more robust are also quite helpful to sort out whether a new predictor genuinely improves the model's ability to predict $Y$.

## 6.6 Inference about the Population Regression Function

We now have established the tools necessary to make inferences about the PRF.

From the bivariate regression, recall the estimated variances of the slope and intercept. Note the "hat" notation $\hat{()}$ – we use hats throughout to signify that a quantity is *estimated* from the sample rather than known from the population. So $\hat{\sigma^2}$ is our estimate of the error variance, and $\hat{var}(b)$ is the estimated sampling variance of the slope coefficient. The same notation applies to $\hat{b}$, which is the estimated population slope coefficient.

$$\begin{aligned} \hat{var}(b) &= \frac{\hat{\sigma^2}}{\sum x_i^2} \\ \hat{var}(a) &= \frac{\hat{\sigma^2} \sum x_i^2}{n \sum x_i^2} \end{aligned}$$

Then, construct a $100(1-\alpha)\%$ confidence interval:

$$\beta = b \pm t_{\alpha/2} SE(b)$$

Locate $t_{\alpha/2}$ using the student's t-distribution with $n-k-1$ degrees of freedom ($k$ corresponds to number of IVs). The procedure is **identical** to t-testing in the single variable case. Instead, our goal is inference about the PRF from the SRF.

It's useful to think about how this logic and procedure is essentially identical to what you encountered in POL 681 regarding a single variable (a mean, and an inference about the population mean; remember the CLT). Likewise, we might conduct the **Null Hypothesis Test**, where for instance:

$$H_a : \beta_k < 0$$

$$H_0 : \beta_k \geq 0$$

We use the standard error to generate a confidence interval and determine whether to reject or retain the null.

As such, we could conduct one or two tailed hypothesis tests. If our expectation is that $\beta$ is positive, for instance, we might formulate:

$$H_a : \beta_k > 0$$

$$H_0 : \beta_k \leq 0$$

If our expectation is that $\beta$ is negative, an inverse relationship:

$$H_a : \beta_k < 0$$

$$H_0 : \beta_k \geq 0$$

Or, perhaps we cannot posit one way or another, and construct a two-tailed test.

$$H_a : \beta_k \neq 0$$

$$H_0 : \beta_k = 0$$

The t-statistic is distributed with $n-k-1$ degrees of freedom. If the computed value falls in the critical region, reject the null.

## 6.7 Additional Tests

The steps required to calculate hypothesis tests and confidence intervals are the same as with a single variable. We could test $H_0 : \beta_1 = \beta_2 .... \beta_k$, by comparing the $R^2$ of nested models.

*Model 1:* $Y_i = \alpha + \beta_1 X_1 + \epsilon_i$

*Model 2:* $Y_i = \alpha + \beta_1 X_1 + \beta_2 X_2 + \epsilon_i$

And model 2, $Y_i = \alpha + \beta_1 X_1 + \beta_2 X_2 + \epsilon_i$. The first model is nested within the second model, because $\beta_2$ is assumed to be 0.

$$\begin{aligned} F &= \frac{RegSS_1 - RegSS_2/q}{RSS_1/n-k-1}, where \\ & F \sim F[q, n-k-1] \end{aligned}$$

## 6.8 Summary

This chapter reviewed the foundations of inference about the population regression function. In regression analysis, it is important to understand the characteristics of the variables that constitute the model. How variables are scaled influence the interpretation of the coefficients. The additive model – with its intercept shifts and parallel slopes – is a powerful starting point, but it assumes the effect of every predictor is constant across groups. In the next chapter, we relax this assumption by introducing **interactions**, which allow slopes to differ across groups or to vary with the level of another predictor.

# Part II

# Part II: Diagnostics and Extensions

# Chapter 7

# Interactions and Non-Additivity

## 7.1 Overview

Up to this point, we've assumed that the linear regression model is **additive** – the effect of one predictor on $Y$ doesn't depend on the value of another predictor. But this assumption is often unrealistic. Does the effect of authoritarianism on institutional trust differ across partisan groups? Does the relationship between income and vote margins change depending on a community's racial composition? These are questions about **interactions**.

In this chapter we develop the interaction model, show how it generalizes the additive (intercept-shift) model from the previous chapter, and apply it to both the Western States Survey and the Arizona precinct data.

## 7.2 The Arizona Precinct Data

Throughout this chapter and the next, we use the **Arizona precinct data** – 1,688 precincts from the 2024 presidential election, with voter registration, turnout, and Census demographics linked at the tract level.

Table 7.1

```
load("precinct_voter_summary.rda")
load("precinct_tract_data.rda")

# Convert raw ACS counts to percentages
precinct_voter_summary <- precinct_voter_summary |>
  dplyr::mutate(
    pct_latino = (tract_acs_latino / tract_acs_total_population) * 100,
    pct_white  = (tract_acs_non_latino_white / tract_acs_total_population) * 100
  )

head(precinct_voter_summary[, c("dos_precinct_key", "trump_harris_margin",
                                "tract_acs_median_household_income",
                                "pct_latino", "tract_acs_median_age",
                                "tract_acs_gini_index")])
```

```
# A tibble: 6 x 6
  dos_precinct_key trump_harris_margin tract_acs_median_household_i~1 pct_latino
  <chr>                          <dbl>                          <dbl>      <dbl>
1 0001 ACACIA                   0.0430                          69005       31.4
2 0002 ACOMA                    0.254                           95395       21.3
3 0003 ACUNA                   -0.508                           49849       93.7
4 0004 ADOBE                    0.115                           78531       30.8
5 0005 ADORA                    0.194                          156354       15.5
6 0006 AGRITOPIA                0.0943                         101179       15.7
# i abbreviated name: 1: tract_acs_median_household_income
# i 2 more variables: tract_acs_median_age <dbl>, tract_acs_gini_index <dbl>
```

**Data: Arizona Precincts**

This data comes from the Arizona Secretary of State voter file, merged with ACS tract-level demographics. Each row is a precinct. The dependent variable is Trump's margin over Harris (positive = Trump advantage). Predictors come from ACS tract-level demographics: median household income, Latino population share, median age, and the Gini index of income inequality.

**dplyr**

The code above uses several `dplyr` verbs:

- `mutate()` creates new columns (or modifies existing ones). Here we divide the raw Latino count by total population to get a percentage.
- `select()` chooses specific columns for subsetting.
- `filter()` keeps only rows that meet a condition (e.g., `filter(!is.na(trump_harris_margin))`).
- `group_by()` + `summarize()` computes summary statistics within groups.
- The pipe `|>` passes the result of one step as the first argument to the next.

Figure 7.1 shows the geographic distribution of Trump's margin across Arizona precincts.

A baseline additive model: can tract-level ACS demographics predict the Trump-Harris margin?

## 7.3 Additive to Interactive

### 7.3.1 The Additive Model

In the previous chapter, we saw how dummy variables produce **intercept shifts** – parallel lines with different baselines. The additive model assumes the slope is the same for every group. For the Western States Survey data:

$$Y_i = \alpha + \gamma_{\text{Rep}} D_{\text{Rep}} + \gamma_{\text{Dem}} D_{\text{Dem}} + \beta_{\text{Auth}} X_{\text{Auth}} + e_i$$

### 7.3.2 The Interaction Model

Recall the additive model assumes the effect of authoritarianism on institutional trust is the **same** for Republicans, Democrats, and Independents. The intercept shifts, but the slope doesn't. What if the effect of authoritarianism *differs* across party groups? The additive model cannot capture this:

$$\begin{aligned} E(Y_i \mid \text{Republican}) &= (\alpha + \gamma_{\text{Rep}}) + \beta_{\text{Auth}} X_{\text{Auth}} \\ E(Y_i \mid \text{Democrat}) &= (\alpha + \gamma_{\text{Dem}}) + \beta_{\text{Auth}} X_{\text{Auth}} \\ E(Y_i \mid \text{Independent}) &= \alpha + \beta_{\text{Auth}} X_{\text{Auth}} \end{aligned}$$

Here, $\frac{\partial Y}{\partial X_{\text{Auth}}} = \beta_{\text{Auth}}$ . We can relax this by estimating an *interactive* model that allows the slope on authoritarianism to differ by party:

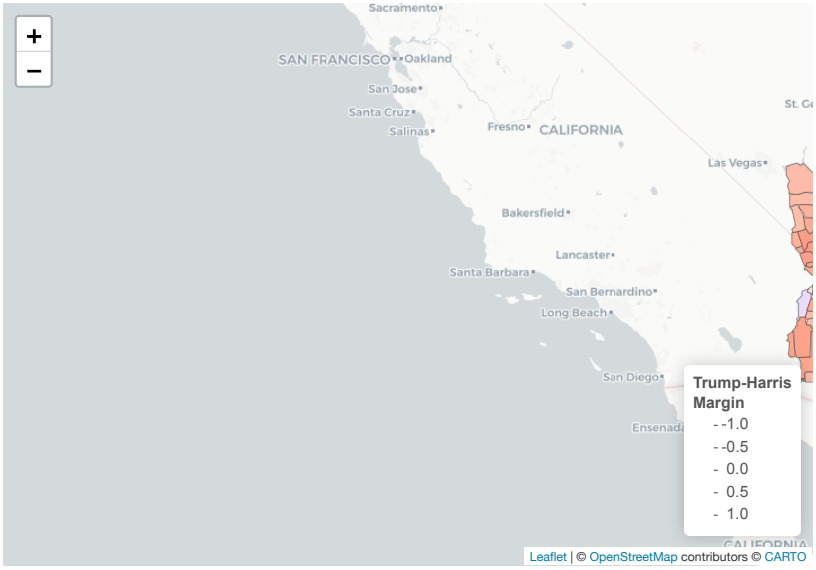



Figure 7.1: Arizona precincts colored by Trump-Harris margin. Hover for precinct details.

Table 7.2

```
fit_az <- lm(trump_harris_margin ~ tract_acs_median_household_income +
               pct_latino + tract_acs_median_age + tract_acs_gini_index,
             data = precinct_voter_summary)
summary(fit_az)
```

```
Call:
lm(formula = trump_harris_margin ~ tract_acs_median_household_income +
    pct_latino + tract_acs_median_age + tract_acs_gini_index,
    data = precinct_voter_summary)

Residuals:
    Min      1Q  Median      3Q     Max
-1.0910 -0.1948 -0.0197  0.1774  1.1668

Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                      1.567e-02  6.626e-
02   0.236    0.813
tract_acs_median_household_income  1.213e-06  2.187e-
07   5.545 3.40e-08 ***
pct_latino                       -1.575e-03  3.810e-04  -
4.134 3.75e-05 ***
tract_acs_median_age              1.048e-02  6.768e-
04  15.490  < 2e-16 ***
tract_acs_gini_index             -1.199e+00  1.115e-01 -
10.756  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2896 on 1674 degrees of freedom
  (9 observations deleted due to missingness)
Multiple R-squared:  0.2631,    Adjusted R-squared:  0.2613
F-statistic: 149.4 on 4 and 1674 DF,  p-value: < 2.2e-16
```

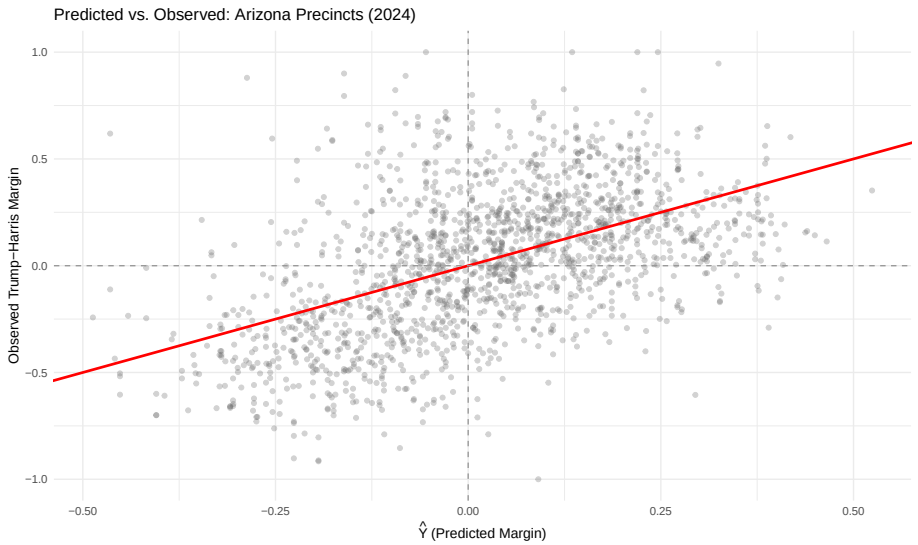



Figure 7.2: Predicted vs. observed Trump-Harris margin across Arizona precincts.

$$Y_i = \alpha + \gamma_{\text{Rep}} D_{\text{Rep}} + \gamma_{\text{Dem}} D_{\text{Dem}} + \beta_{\text{Auth}} X_{\text{Auth}} + \delta_{\text{Rep}} (X_{\text{Auth}} \times D_{\text{Rep}}) + \delta_{\text{Dem}} (X_{\text{Auth}} \times D_{\text{Dem}}) + e_i$$

So what we're doing is just creating two additional variables. One is defined by multiplying authoritarianism by the Republican dummy, and the other is defined by multiplying authoritarianism by the Democrat dummy.

We now include these in the model.

$$\begin{aligned} E(Y_i \mid \text{Independent}) &= \alpha + \beta_{\text{Auth}} X_{\text{Auth}} \\ E(Y_i \mid \text{Republican}) &= (\alpha + \gamma_{\text{Rep}}) + (\beta_{\text{Auth}} + \delta_{\text{Rep}}) X_{\text{Auth}} \\ E(Y_i \mid \text{Democrat}) &= (\alpha + \gamma_{\text{Dem}}) + (\beta_{\text{Auth}} + \delta_{\text{Dem}}) X_{\text{Auth}} \end{aligned}$$

But note what happens when we take the derivative with respect to authoritarianism. The marginal effect of authoritarianism now depends on party identification:

- $\beta_{\text{Auth}}$ is the slope of authoritarianism for **Independents** (the reference group)
- $\delta_{\text{Rep}}$ is the **difference** in the authoritarianism slope between Republicans and Independents
- $\delta_{\text{Dem}}$ is the **difference** in the authoritarianism slope between Democrats and Independents

- $\gamma_{\text{Rep}}$ and $\gamma_{\text{Dem}}$ are the intercept differences when $X_{\text{Auth}} = 0$

The lines are no longer parallel – each group gets its own intercept *and* its own slope.

And note that the marginal effect of authoritarianism is now a function of party identification:

$$\frac{\partial Y}{\partial X_{\text{Auth}}} = \beta_{\text{Auth}} + \delta_{\text{Rep}} D_{\text{Rep}} + \delta_{\text{Dem}} D_{\text{Dem}}$$

**Always Remember to Include Lower-Order Terms**

*It's important to always* include the lower-order constituent terms in an interactive model.

$$Y_i = \alpha + \beta_{\text{Auth}} X_{\text{Auth}} + \delta_{\text{Rep}}(X_{\text{Auth}} \times D_{\text{Rep}}) + \delta_{\text{Dem}}(X_{\text{Auth}} \times D_{\text{Dem}}) + e_i$$

This omits $\gamma_{\text{Rep}}$ and $\gamma_{\text{Dem}}$, forcing the assumption that all three groups share the same intercept (i.e., no group differences when authoritarianism = 0). Conversely:

$$Y_i = \alpha + \gamma_{\text{Rep}} D_{\text{Rep}} + \gamma_{\text{Dem}} D_{\text{Dem}} + \delta_{\text{Rep}}(X_{\text{Auth}} \times D_{\text{Rep}}) + \delta_{\text{Dem}}(X_{\text{Auth}} \times D_{\text{Dem}}) + e_i$$

This omits $\beta_{\text{Auth}}$, forcing the slope to be zero for Independents. We can always *test* whether $\gamma_{\text{Rep}} = \gamma_{\text{Dem}} = 0$ or $\beta_{\text{Auth}} = 0$, but the full model should include all constituent terms.

Compare Figure 7.3 to the additive model from the previous chapter – the lines are no longer parallel. The F-test tells us whether the improvement in fit from allowing different slopes is statistically meaningful.

## 7.4 Continuous-by-Continuous Interactions

The interaction we just estimated involved a continuous variable (authoritarianism) and a categorical variable (party ID). But interactions are not limited to this combination. We can also interact two continuous variables. The logic is the same – the effect of one variable depends on the value of another – but the interpretation requires more care because neither variable "turns on and off" like a dummy does.

Consider the Arizona precinct data. We might ask: does the effect of income on Trump's margin depend on the community's Latino population share? In a predominantly white precinct, higher income might push margins in one direction; in a predominantly Latino precinct, the income effect might be different – or even reversed.

Table 7.3

```r
# Additive model (restricted)
fit_additive <- lm(institutional_trust ~ authoritarianism + republican + democrat, data

# Interactive model (unrestricted)
fit_interaction <- lm(institutional_trust ~ authoritarianism * republican + authoritari
summary(fit_interaction)
```

```
Call:
lm(formula = institutional_trust ~ authoritarianism * republican +
    authoritarianism * democrat, data = wss20)

Residuals:
     Min       1Q   Median       3Q      Max
-0.58654 -0.10874  0.00852  0.12260  0.58245

Coefficients:
                             Estimate Std. Error t value Pr(>|t|)
(Intercept)                   0.40529    0.01215  33.348   <2e-
16 ***
authoritarianism              0.04905    0.02221   2.208   0.0273 *
republican                    0.16678    0.01637  10.188   <2e-
16 ***
democrat                      0.03461    0.01379   2.510   0.0121 *
authoritarianism:republican -0.03458    0.02819  -1.227   0.2201
authoritarianism:democrat     0.02635    0.02579   1.021   0.3071
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1835 on 3425 degrees of freedom
  (169 observations deleted due to missingness)
Multiple R-squared:  0.1072,    Adjusted R-squared:  0.1059
F-statistic: 82.25 on 5 and 3425 DF,  p-value: < 2.2e-16
```

```r
# F-test: do the interaction terms jointly improve the model?
anova(fit_additive, fit_interaction)
```

```
Analysis of Variance Table

Model 1: institutional_trust ~ authoritarianism + republican + democrat
Model 2: institutional_trust ~ authoritarianism * republican + authoritarianism *
    democrat
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1   3427 115.57
2   3425 115.30  2   0.26505 3.9366 0.0196 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

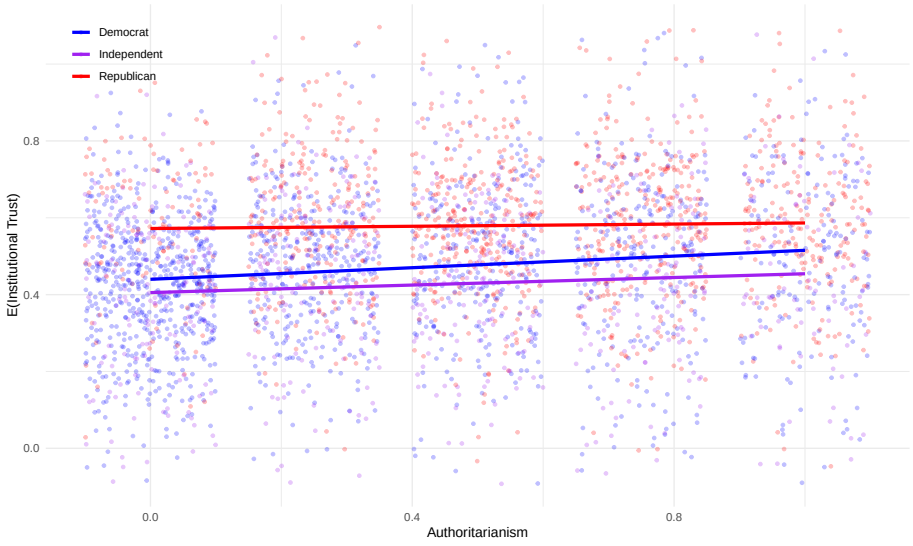



Figure 7.3: Interaction between authoritarianism and party identification. Unlike the additive model, slopes now differ across groups.

The model is:

$$\text{Margin}_i = \beta_0 + \beta_1 \text{Income}_i + \beta_2 \text{Latino}_i + \beta_3 (\text{Income}_i \times \text{Latino}_i) + \beta_4 \text{Age}_i + \beta_5 \text{Gini}_i + e_i$$

Now, what is the effect of income on the margin? Take the partial derivative with respect to income:

$$\frac{\partial \text{Margin}}{\partial \text{Income}} = \beta_1 + \beta_3 \times \text{Latino}_i$$

This is the key insight: the marginal effect of income is *not* a single number. It's a function of percent Latino. At a precinct where Latino = 0, the effect of income is just $\beta_1$. At a precinct where Latino = 50, it's $\beta_1 + 50\beta_3$. The interaction coefficient $\beta_3$ tells you how much the income effect *changes* for each one-unit increase in percent Latino – but $\beta_3$ by itself doesn't tell you what the income effect actually *is* at any particular value of percent Latino. You need both $\beta_1$ and $\beta_3$ together.

This is why you can't just look at the coefficient table and say "the interaction is significant, so the effect depends on Latino population." You need to compute the marginal effect at specific values and plot it. Let's do that.

Table 7.4

```
fit_interact_az <- lm(trump_harris_margin ~ tract_acs_median_household_income *
                        pct_latino + tract_acs_median_age + tract_acs_gini_index,
                      data = precinct_voter_summary)
summary(fit_interact_az)
```

```
Call:
lm(formula = trump_harris_margin ~ tract_acs_median_household_income *
    pct_latino + tract_acs_median_age + tract_acs_gini_index,
    data = precinct_voter_summary)

Residuals:
     Min       1Q   Median       3Q      Max
-1.09183 -0.19607 -0.02004  0.17629  1.16503

Coefficients:
                                             Estimate Std. Error t value
(Intercept)                                 2.389e-02  6.700e-
02   0.357
tract_acs_median_household_income            1.365e-06  2.850e-
07   4.790
pct_latino                                  -9.457e-04  8.452e-
04  -1.119
tract_acs_median_age                         1.040e-02  6.839e-
04  15.210
tract_acs_gini_index                        -1.231e+00  1.179e-
01 -10.443
tract_acs_median_household_income:pct_latino -1.057e-08  1.267e-
08  -0.834
                                            Pr(>|t|)
(Intercept)                                    0.721
tract_acs_median_household_income           1.81e-06 ***
pct_latino                                     0.263
tract_acs_median_age                         < 2e-16 ***
tract_acs_gini_index                         < 2e-16 ***
tract_acs_median_household_income:pct_latino    0.404
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2896 on 1673 degrees of freedom
  (9 observations deleted due to missingness)
Multiple R-squared:  0.2634,    Adjusted R-squared:  0.2612
F-statistic: 119.6 on 5 and 1673 DF,  p-value: < 2.2e-16
```

### 7.4.1 Computing Marginal Effects

The marginal effect of income at any value of percent Latino is just the partial derivative we wrote above:

$$\frac{\partial \text{Margin}}{\partial \text{Income}} = \hat{\beta}_1 + \hat{\beta}_3 \times \text{Latino}$$

We plug in different values of percent Latino, compute the marginal effect at each one, and plot it.

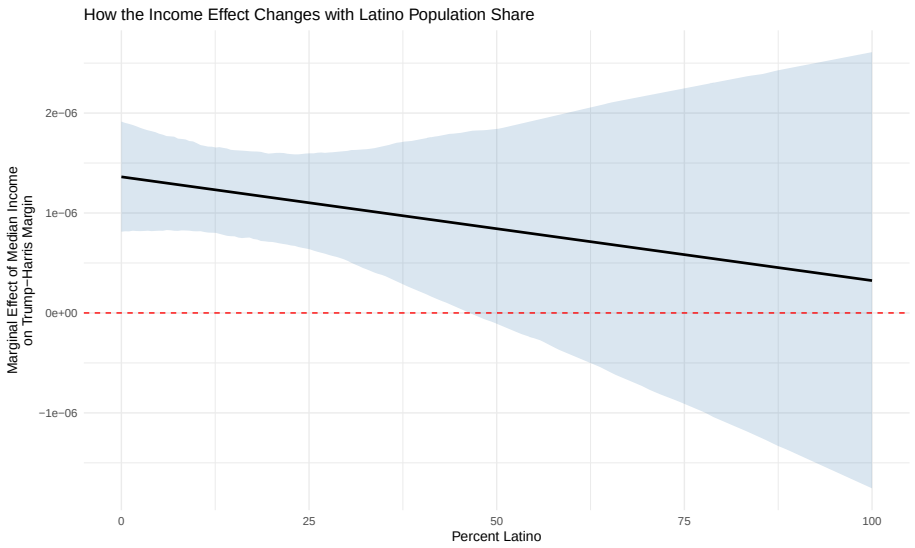



Figure 7.4: Marginal effect of median household income on Trump-Harris margin at different levels of percent Latino. Shaded region is the 95% confidence interval. The red dashed line at zero indicates no effect.

How do we read Figure 7.4? The black line is the estimated marginal effect of income at each level of percent Latino. The shaded band is the 95% confidence interval. Where the band includes zero (the red dashed line), we cannot say the income effect is statistically distinguishable from zero at that level of Latino population. Where the band is entirely above or below zero, the income effect is statistically significant at the 5% level.

This is the right way to interpret a continuous interaction. The coefficient on the interaction term alone – $\hat{\beta}_3$ – tells you the *rate of change* of the income effect as Latino percentage increases. But it doesn't tell you what the income effect *is* at any particular value. You always need the full marginal effect calculation and the plot.

## 7.5 Summary

This chapter extended the additive regression model to allow **interactions** – cases where the effect of one predictor depends on the value of another. Key takeaways:

- **Categorical interactions** allow different slopes for different groups (e.g., the effect of authoritarianism differs by party).
- **Continuous interactions** allow the marginal effect of one variable to vary smoothly with another (e.g., the income effect changes with Latino population share).
- Always include the lower-order constituent terms when specifying an interaction.
- Use **marginal effect plots** to interpret continuous interactions – the coefficient on the interaction term alone is not sufficient.
- The **F-test** (`anova()`) can formally test whether interaction terms jointly improve the model over the restricted additive specification.

In the next chapter, we examine what happens when a core assumption of the linear model – constant error variance – is violated. The Arizona precinct data will turn out to be a natural setting for this problem.

# Chapter 8

# Heteroskedasticity and Weighted Least Squares

## 8.1 Data and Setup

We'll continue with the **Arizona precinct data** introduced in the previous chapter. This is a natural setting for heteroskedasticity – precincts vary enormously in size, from a few hundred voters to tens of thousands, and we'd expect the variance of election margins to depend on precinct characteristics.

## 8.2 Recall the Gauss Markov Theorem

Now, let's explore the issues that emerge with heteroskedasticity. To understand the consequences, it's useful to recall the Gauss-Markov theorem and its assumptions. The G-M theorem states that under certain conditions, the OLS estimator is the Best Linear Unbiased Estimator (BLUE). The key assumptions

Table 8.1

```
load("precinct_voter_summary.rda")
load("precinct_tract_data.rda")

# Convert raw counts to percentages (as in the previous chapter)
precinct_voter_summary <- precinct_voter_summary |>
  dplyr::mutate(
    pct_latino = (tract_acs_latino / tract_acs_total_population) * 100,
    pct_white  = (tract_acs_non_latino_white / tract_acs_total_population) * 100
  )
```

are summarized in Table 8.2.

Table 8.2: The Nine Gauss-Markov Assumptions

| # | Assumption | Mathematical Representation | Purpose/Implications |
|---|---|---|---|
| 1 | **Linearity** | $y_i = \mathbf{x}_i^T\beta + \epsilon_i$ | Estimation / Unbiasedness |
| 2 | **Exogenous (fixed) IVs** | $\text{cov}(X_i, \epsilon_i) = 0$ | Estimation / Unbiasedness |
| 3 | **Zero mean error** | $E(\epsilon_i) = 0$ | Inference |
| 4 | **Homoskedasticity** | $\text{var}(\epsilon_i) = \sigma^2$ for all $i$ | Inference |
| 5 | **Normality** | $\epsilon_i \sim N(0, \sigma)$ | Inference |
| 6 | **Independent errors** | $\text{cov}(\epsilon_i, \epsilon_j) = 0$ for $i \neq j$ | Inference |
| 7 | **Variation in X** | $\text{var}(X) > 0$ | Estimation |
| 8 | **Correct specification** | No omitted variables, no extraneous variables | Estimation |
| 9 | **No perfect multicollinearity** | No predictor is a perfect linear combination of others | Estimation |

Assumptions 1–2 and 7–9 are needed for OLS to produce **unbiased** coefficient estimates. Assumptions 3–6 are needed for valid **inference** – hypothesis tests and confidence intervals. In this chapter, we focus on what happens when Assumption 4 (homoskedasticity) is violated.

If $\mathbf{b}$ is a linear function of $\mathbf{y}$, then $E(\mathbf{b}) = \beta$. This is just the matrix version of the unbiasedness property of the OLS estimator. To see this, we can write $\mathbf{b}$ as a function of $\mathbf{y}$:

$$\mathbf{b} = (\mathbf{X^T X})^{-1}(\mathbf{X^T X}\beta + \epsilon)$$

Which is,

$$(\mathbf{X^T X})^{-1}\mathbf{X^T X}\beta + (\mathbf{X^T X})^{-1}\mathbf{X^T}\epsilon$$

By taking expectations, $E(\mathbf{b}) = \beta$.

As in scalar form, the minimum variance component is the most involved. We need to show that the OLS estimator has the *smallest* variance among all linear

unbiased estimators – this is what makes it "best." We start by specifying a variance-covariance matrix for $\mathbf{b}$.

$$\text{var-cov}(\mathbf{b}) = E[(\mathbf{b} - \beta)(\mathbf{b} - \beta)^T]$$

Substituting $\mathbf{b} - \beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\epsilon$ and expanding:

$$E[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\epsilon\epsilon^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}]$$

Since $E(\epsilon\epsilon^T) = \sigma^2\mathbf{I}$ under the Gauss-Markov assumptions (homoskedastic, uncorrelated errors), we can pull $\sigma^2\mathbf{I}$ out and simplify:

$$(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T \cdot \sigma^2\mathbf{I} \cdot \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1} = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1} = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}$$

The middle term $\mathbf{X}^T\mathbf{X}$ cancels with one of the inverses, leaving us with $\sigma^2(\mathbf{X}^T\mathbf{X})^{-1}$. This is the variance-covariance matrix of the OLS estimator. What does it tell us?

- The **diagonal entries** give the variance of each coefficient: $\text{var}(b_k) = \sigma^2[(\mathbf{X}^T\mathbf{X})^{-1}]_{kk}$. Taking the square root gives the standard error. A larger diagonal entry means less precision – the estimate is noisier.
- The **off-diagonal entries** give the covariance between pairs of coefficients: $\text{cov}(b_j, b_k) = \sigma^2[(\mathbf{X}^T\mathbf{X})^{-1}]_{jk}$. When predictors are correlated, this covariance grows, meaning the estimates "move together" – knowing that one coefficient is too high tells you the other is likely too low (or vice versa).
- The entire matrix depends on $\mathbf{X}^T\mathbf{X}$, which is the cross-product of the design matrix. More data (larger $n$) makes $\mathbf{X}^T\mathbf{X}$ larger, its inverse smaller, and the variances shrink. More collinear predictors inflate $(\mathbf{X}^T\mathbf{X})^{-1}$, making the variances blow up – this is the VIF problem.

> ⚠ **Why This Matters**
>
> The simplification *only works* because $E(\epsilon\epsilon^T) = \sigma^2\mathbf{I}$. If the errors are heteroskedastic – $E(\epsilon\epsilon^T) = \sigma^2\Omega$ – then the $\Omega$ sits in the middle of the formula things don't simplify. The variance becomes a much longer expression, and the OLS formula $\sigma^2(\mathbf{X}^T\mathbf{X})^{-1}$ is incorrect. The OLS estimator is unbiased, but **does not have minimum variance** – it's not BLUE.

### 8.2.1 An Alternative?

Recall, that to show minimum variance, we suppose there exists *any* other linear unbiased estimator $\tilde{\mathbf{b}} = \mathbf{C}\mathbf{y}$ where $\mathbf{C}$ is some matrix (not necessarily $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$). Remember, there are numerous ways to construct a linear estimator – we've been following the objective function that the linear estimator should minimize the sum of squared residuals.

Instead, let's assume $\mathbf{C} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T + \mathbf{L}$, where $\mathbf{L}$ captures how $\tilde{\mathbf{b}}$ differs from OLS.

$$\text{var}(\tilde{\mathbf{b}}) = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1} + \sigma^2\mathbf{L}\mathbf{L}^T$$

The term $\sigma^2\mathbf{L}\mathbf{L}^T$ is estimable – LL can be inverted, so $\text{var}(\tilde{\mathbf{b}}) \geq \text{var}(\mathbf{b})$. The OLS estimator achieves the smallest variance – it is *best*.

The variance-covariance matrix for $\mathbf{b}$ is:

$$\text{var}(\mathbf{b}) = E[(\mathbf{b} - \beta)(\mathbf{b} - \beta)^T]$$

$$E(\epsilon\epsilon^T) = \sigma^2\mathbf{I}$$

Where we may use $\hat{\sigma}^2$ when $\sigma^2$ is not known.

Then, $\mathbf{y} \sim N(\mathbf{X}\mathbf{b}, \sigma^2\mathbf{I})$.

And $\text{var}(\epsilon_i) = \sigma^2 \ \forall\, i$. **Every respondent has the same error component and zero covariance with all other responses.**

$$\begin{bmatrix} \sigma^2 & \cdots & 0 \\ 0 & \sigma^2 & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & \sigma^2 \end{bmatrix}$$

## 8.3 Heteroskedasticity

If the variance is not constant, then the OLS estimator is not BLUE. It will be unbiased, but not efficient.

The estimator under the Gaussian assumptions is **unbiased**, but not efficient (thus not BLUE). As a consequence, inferences will be quite imprecise, why? The key thing to remember is this:

$$\begin{bmatrix} \sigma^2 & \cdots & 0 \\ 0 & \sigma^2 & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & \sigma^2 \end{bmatrix}$$

It encapsulates the properties of the error term, for every unit.Constant diagonal elements mean that the variance is constant; off-diagonal zeros mean that the errors are uncorrelated. If either of these conditions fails, the OLS estimator is no longer BLUE.

- **Units (respondents) are heterogeneous.** Consider a model predicting political contributions from income. Low-income individuals cluster tightly near zero, but high-income individuals vary quite a lot – some donate thousands, others nothing. The variance of contributions fans out as income increases. It looks like a "megaphone" when plotted against income.
- **Outliers exist.** Sometimes we observe extreme cases; outliers; a handful of extreme observations. Imagine in our voting precinct data, a few precincts with different demographic profiles or spikes in turnout (maybe some politically active precincts vote at high rates). This can inflate the residual variance in certain regions of the predictor space while leaving it small elsewhere.
- **Models are incorrectly specified.** If the true relationship is nonlinear (e.g., imagine the nonlinear relationship between age and turnout) but we fit a linear model, the residuals will be systematically larger where the misspecification is worst. The model looks homoskedastic in some regions, but heteroskedastic in others.
- **Learning occurs.** In panel or repeated-measures designs, respondents may become more consistent over time. Early survey waves show high variability in responses as people form opinions, but later waves converge. The variance of the error term shrinks across time periods.

## 8.4 Building the Weighted Least Squares Estimator

Let's start with this: $E(\epsilon\epsilon^T) = \sigma^2\Omega$

If this omega matrix is just the identity matrix – no heteroskedasticity, no autocorrelation – then $\Omega = \mathbf{I}$, and the estimator is BLUE.

But if not, then the var($\mathbf{b}$) is wrong, because $E(\epsilon\epsilon^T) \neq \sigma^2\mathbf{I}$.

Instead, the var($\mathbf{b}$) is

$$\sigma^2(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\Omega\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}$$

And:

$$\text{var}(\mathbf{b}) \neq \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}$$

Only if $\Omega = \mathbf{I}$ will the variance estimate be correct; otherwise, we'll either under and/or over-estimate the variance. This results in several important consequences:

- Wrong t-values
- Incorrect statistical tests
- Wrong, imprecise estimates
- Larger confidence intervals
- Type II Error

**The Key Point**

If $\Omega$ is $\mathbf{I}$ then this reduces to the OLS estimator of the variance; if it doesn't, the term doesn't simplify, the variance is not $\sigma^2(\mathbf{X}^T\mathbf{X})^{-1}$, and the OLS estimator is not BLUE. The OLS estimator is unbiased, but inefficient. The SEs, Confidence Intervals, p-values, etc. are **WRONG**! Inference is suspect. Hypothesis testing won't be accurate.

**Where Do Weights Come From? Common Weighting Schemes**

WLS and GLS are not just tools for fixing heteroskedasticity – they appear whenever observations carry unequal information. The `weights` argument in R's `lm()` expects $w_i = 1/\text{var}(\epsilon_i)$: observations with *larger* weights count *more*.

**Frequency weights.** When a single row in your dataset represents multiple identical units, the weight is the count. A row representing 500 households gets 500 times the influence of a row representing 1. Weights can be frequency weights.

**Probability (sampling) weights.** In survey designs, respondents are sampled with unequal probabilities $\pi_i$. The weight $w_i = 1/\pi_i$ corrects for this: a respondent sampled with probability 0.01 (representing 100 people in the population) gets weight 100 – `svydesign()` specifies the sampling design and `svyglm()` produces design-consistent estimates. Most large-scale surveys (ANES, CPS, ACS) provide sampling weights in the data.

**Propensity score weights (causal inference).** In observational studies, inverse probability of treatment weighting (IPTW) uses the propensity score $e(\mathbf{x}_i) = P(\text{treated} \mid \mathbf{x}_i)$ to construct weights that balance treatment and control groups [Rosenbaum and Rubin, 1983]. Treated units get weight $1/e(\mathbf{x}_i)$ and control units get weight $1/(1-e(\mathbf{x}_i))$. The resulting weighted regression estimates the average treatment effect as if treatment had been randomly assigned. This connects WLS to the broader causal inference toolkit – the weights don't correct for heteroskedasticity per se, but they use the same mathematical machinery of weighted estimation.

### 8.4.1 Modeling Variation

The core insight behind GLS/WLS is that the errors need not be constant. Instead of assuming constant variance, transform the data in such a way that the *transformed data have constant variance. Then* apply OLS to the transformed data.

The transformed OLS estimator will be BLUE.

Recall the problem is $E(\epsilon\epsilon^T) = \sigma^2\Omega$, where $\Omega \neq \mathbf{I}$.

The diagonal entries of $\Omega$ tell us how much each observation's error variance deviates from the average – some observations are "noisier" than others. Some observations fall far from the regression line. Our goal is to find a transformation that produces new errors $\epsilon^*$ – how can we change the data to restore the property we need:

$$E(\epsilon^*\epsilon^{*T}) = \sigma^2\mathbf{I}$$

Think of it this way: if observation $i$ has a large error variance (it's noisy, unreliable), we want to *downweight* it – make it count less in the estimation. If observation $j$ has a small error variance (it's precise, reliable), we want to *upweight* it. Of course, the question is, how do we calculate these weights?

- **Specify the weight matrix.** We need $\Omega$, the matrix that describes how the error variance differs across observations, as well as its inverse $\Omega^{-1}$. **The diagonal entries $\omega_{ii}$ represent the relative variance of each observation.** For instance, a precinct with $\omega_{ii} = 4$ has four times the error variance of the baseline.

- **Decompose the inverse.** Define $\rho$ such that $\rho^T\rho = \Omega^{-1}$.

  Why do we need this step? Recall what we're trying to do: transform the regression equation so that the errors become homoskedastic. In scalar algebra, if the variance of $\epsilon_i$ is $\sigma^2\omega_{ii}$ instead of $\sigma^2$, we'd just divide the entire equation by $\sqrt{\omega_{ii}}$. Basically, we're multiplying the regression variance by a weight that adjusts for the heteroskedasticity.

  The transformed error $\epsilon_i^* = \epsilon_i/\sqrt{\omega_{ii}}$ would then have variance $\sigma^2\omega_{ii}/\omega_{ii} = \sigma^2$ – problem solved.

  But in matrix form, we can't "divide by" $\Omega$. Matrix division doesn't exist the way scalar division does. What we *can* do is premultiply by a matrix. So we need a matrix $\rho$ that, when we premultiply the regression equation by it, has the same effect as dividing each observation by the square root of its variance weight.

  The requirement is that $\rho$ "undoes" $\Omega$ in the error covariance. Start from the transformed errors: $\epsilon^* = \rho\epsilon$. Their covariance is:

$$E(\epsilon^*\epsilon^{*T}) = E(\rho\epsilon\epsilon^T\rho^T) = \rho\, E(\epsilon\epsilon^T)\, \rho^T = \rho\,(\sigma^2\Omega)\,\rho^T = \sigma^2\rho\Omega\rho^T$$

  For this to equal $\sigma^2\mathbf{I}$, we need $\rho\Omega\rho^T = \mathbf{I}$, which means $\rho^T\rho = \Omega^{-1}$.

  Since $\Omega$ is diagonal (no correlation between errors – just unequal variances), the decomposition is simple. Each diagonal entry of $\rho$ is:

$$\rho_{ii} = \frac{1}{\sqrt{\omega_{ii}}}$$

  A concrete example: suppose we have three observations with $\omega_{11} = 1$, $\omega_{22} = 4$, $\omega_{33} = 9$. Then:

$$\Omega = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 9 \end{bmatrix}, \quad \rho = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/3 \end{bmatrix}$$

Observation 1 (low variance, $\omega = 1$) gets multiplied by 1 – no change. Observation 2 (moderate variance, $\omega = 4$) gets multiplied by 1/2 – halved. Observation 3 (high variance, $\omega = 9$) gets multiplied by 1/3 – shrunk the most. Noisy observations are downweighted; precise observations are left intact.

You can check this out for yourself. Just verify that $\rho^T\rho = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/4 & 0 \\ 0 & 0 & 1/9 \end{bmatrix} = \Omega^{-1}$. And you could then easily show $\Omega\Omega^{-1} = \mathbf{I}$.

- **Premultiply and estimate.** Multiply every variable in the regression – both $\mathbf{y}$ and $\mathbf{X}$ – by $\rho$. Then run OLS on the transformed equation. Observations with large variance get multiplied by a small weight (shrunk toward zero), while precise observations get multiplied by a large weight (amplified).

Here, we are effectively weighting the observations according to how much they vary around $\hat{y}_i$. Noisy observations contribute less to the sum of squared residuals; precise observations contribute more. The result is an estimator that extracts the maximum information from each data point.

### 8.4.2 More Elaboration

Okay, so to recap, we want this matrix:

$$\Omega = \begin{bmatrix} w_{11} & \cdots & 0 \\ 0 & w_{22} & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & w_{nn} \end{bmatrix}$$

And we can calculate the inverse,

$$\Omega^{-1} = \begin{bmatrix} 1/w_{11} & \cdots & 0 \\ 0 & 1/w_{22} & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & 1/w_{nn} \end{bmatrix}$$

$$\rho = \begin{bmatrix} 1/\sqrt{w_{11}} & \cdots & 0 \\ 0 & 1/\sqrt{w_{22}} & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & 1/\sqrt{w_{nn}} \end{bmatrix}$$

And, $\rho^T\rho = \Omega^{-1}$.

Now premultiply the entire regression equation $\mathbf{y} = \mathbf{X}\beta + \epsilon$ by $\rho$. We apply $\rho$ to each term separately:

$$\rho \mathbf{y} = \rho \mathbf{X} \beta + \rho \epsilon$$

Let's write out each transformed component.

**Transforming y:**

$$\mathbf{y}^* = \rho \mathbf{y} = \begin{bmatrix} 1/\sqrt{w_{11}} & \cdots & 0 \\ 0 & 1/\sqrt{w_{22}} & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & 1/\sqrt{w_{nn}} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} y_1/\sqrt{w_{11}} \\ y_2/\sqrt{w_{22}} \\ \vdots \\ y_n/\sqrt{w_{nn}} \end{bmatrix}$$

Each $y_i$ gets divided by the square root of its variance weight. Observations with large $w_{ii}$ (noisy) are shrunk; observations with small $w_{ii}$ (precise) are left relatively unchanged.

**Transforming X:**

$$\mathbf{X}^* = \rho \mathbf{X} = \begin{bmatrix} 1/\sqrt{w_{11}} & \cdots & 0 \\ 0 & 1/\sqrt{w_{22}} & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & 1/\sqrt{w_{nn}} \end{bmatrix} \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots \\ 1 & x_{21} & x_{22} & \cdots \\ \vdots & \vdots & \vdots & \ddots \\ 1 & x_{n1} & x_{n2} & \cdots \end{bmatrix} = \begin{bmatrix} 1/\sqrt{w_{11}} & x_{11}/\sqrt{w_{11}} & x_{12}/\sqrt{w_{11}} \\ 1/\sqrt{w_{22}} & x_{21}/\sqrt{w_{22}} & x_{22}/\sqrt{w_{22}} \\ \vdots & \vdots & \vdots \\ 1/\sqrt{w_{nn}} & x_{n1}/\sqrt{w_{nn}} & x_{n2}/\sqrt{w_{nn}} \end{bmatrix}$$

Notice something important: every entry in observation $i$'s row – including the intercept column of 1's – gets divided by $\sqrt{w_{ii}}$. The intercept is no longer a column of ones; it becomes $1/\sqrt{w_{ii}}$. This is why WLS changes the intercept estimate as well.

**Transforming $\epsilon$:**

$$\epsilon^* = \rho \epsilon = \begin{bmatrix} \epsilon_1/\sqrt{w_{11}} \\ \epsilon_2/\sqrt{w_{22}} \\ \vdots \\ \epsilon_n/\sqrt{w_{nn}} \end{bmatrix}$$

And this is the whole point. The variance of $\epsilon_i^*$ is:

$$\text{var}(\epsilon_i^*) = \text{var}\left(\frac{\epsilon_i}{\sqrt{w_{ii}}}\right) = \frac{1}{w_{ii}} \cdot \text{var}(\epsilon_i) = \frac{1}{w_{ii}} \cdot \sigma^2 w_{ii} = \sigma^2$$

The heteroskedasticity is gone. Every transformed error has the same variance $\sigma^2$.

**The transformed equation** is now:

$$\mathbf{y}^* = \mathbf{X}^*\beta + \epsilon^*$$

where $E(\epsilon^*\epsilon^{*T}) = \sigma^2\mathbf{I}$. We can now safely apply OLS to this transformed system. The OLS estimator for the transformed equation is:

$$\mathbf{b}^* = (\mathbf{X}^{*T}\mathbf{X}^*)^{-1}\mathbf{X}^{*T}\mathbf{y}^*$$

Substituting $\mathbf{X}^* = \rho\mathbf{X}$ and $\mathbf{y}^* = \rho\mathbf{y}$:

$$\mathbf{b}^* = ((\rho\mathbf{X})^T(\rho\mathbf{X}))^{-1}(\rho\mathbf{X})^T(\rho\mathbf{y})$$

$$= (\mathbf{X}^T\rho^T\rho\mathbf{X})^{-1}\mathbf{X}^T\rho^T\rho\mathbf{y}$$

Since $\rho^T\rho = \Omega^{-1}$:

$$\mathbf{b}^* = (\mathbf{X}^T\Omega^{-1}\mathbf{X})^{-1}\mathbf{X}^T\Omega^{-1}\mathbf{y}$$

If we were to estimate this model – OLS with weights corresponding to the nature of heteroskedasticity – this produces the *weighted least squares* estimator. The estimator – after the necessary transformations – is BLUE. For instance, $E(\mathbf{b}^*) = \beta$.

## 8.5 Properties

If we can make this adjustment, then $E(\epsilon^*\epsilon^{*T}) = \sigma^2\mathbf{I}$.

Because,

$$E(e_i \cdot e_j) = 0 \quad \forall\, i \neq j$$

And because:

$$E(e_i^2/\omega_i) = (1/\omega_i)E(e_i^2) = (1/\omega_i)\sigma^2\omega_i = \sigma^2$$

Then the WLS estimator is BLUE. The var($\mathbf{b}$) is:

$$\sigma^2(\mathbf{X}^T\Omega^{-1}\mathbf{X})^{-1}\mathbf{X}^T\Omega^{-1}\Omega\Omega^{-1}\mathbf{X}(\mathbf{X}^T\Omega^{-1}\mathbf{X})^{-1}$$

This simplifies to: $\sigma^2(\mathbf{X}^T\Omega^{-1}\mathbf{X})^{-1}$.

## 8.6 Practical Concerns and Limitations

There's a catch with WLS that isn't always made explicit: **to construct the weights, we need to know what's causing the heteroskedasticity.** The entire procedure depends on $\Omega$, but $\Omega$ isn't given to us – we have to specify it. That means we need a theory (or at least a working hypothesis) about *which variable(s)* drive the non-constant variance. Is the error variance proportional to income? To population size? To the square of age? Each assumption produces a different $\Omega$ and therefore a different set of weights.

If we get $\Omega$ right, WLS is BLUE – the best we can do. If we get it wrong, WLS may actually perform *worse* than OLS, because we're weighting observations according to a pattern that doesn't match the true variance structure. We trade one problem (inefficiency from ignoring heteroskedasticity) for another (inefficiency from mis-specifying the weights). This is the fundamental tension with GLS/WLS.

For example, suppose we believe $\sigma_i^2 = \sigma^2 x_{2i}$ – the variance is proportional to $x_2$.

Then,

$$\Omega^{-1} = \begin{bmatrix} 1/x_{21} & \cdots & 0 \\ 0 & 1/x_{22} & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & 1/x_{2n} \end{bmatrix}$$

And,

$$\rho = \begin{bmatrix} 1/\sqrt{x_{21}} & \cdots & 0 \\ 0 & 1/\sqrt{x_{22}} & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & 1/\sqrt{x_{2n}} \end{bmatrix}$$

## 8.7 Some Additional Methods

In many circumstances, we may have little idea about the nature of heteroskedasticity, making it difficult to specify $\Omega$.

An alternative is the method proposed by the economist Halbert White Wikipedia White [1980].

We may estimate $\sigma_i^2$:

$$(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\epsilon\epsilon^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}$$

$$(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\Sigma\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}$$

White [1980] shows that we may estimate $\Sigma$ – i.e., $\hat{\Sigma}$ as,

$$\begin{bmatrix} e_1^2 & \cdots & 0 \\ 0 & e_2^2 & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & e_n^2 \end{bmatrix}$$

## 8.8 Trade-Offs

GLS/WLS is only BLUE under heteroskedasticity when we correctly specify $\Omega$. This makes the method somewhat harder to implement.

White's method allows us to calculate $\Sigma$ as $\hat{\Sigma}$ and use OLS. If in fact $\Sigma = \sigma^2\mathbf{I}$, then White's method will give us the same results as OLS assuming homoskedasticity.

White's method is easier to implement and requires no a priori knowledge about the nature of non-constant variation. Less is known about the small sample properties of both methods (e.g., estimating $\hat{\Sigma}$ from the residuals).

Typically, it is recommended to attempt multiple approaches and examine if they converge on the same solution.

## 8.9 Detection

Let's assume that heteroskedasticity is a function of a single variable, so $\sigma_i^2 = \sigma^2 X_{2i}$.

$$Y_i = \beta_1 + \beta_2 X_{2i} + \beta_3 X_{3i} + \epsilon_i$$

### 8.9.1 Goldfeld-Quandt Test

The Goldfeld-Quandt test proceeds in four steps:

1. Construct the null hypothesis: *no heteroskedasticity.*
2. Sort the data in ascending order of $X_{2i}$.
3. Remove $C$ observations in the middle of the distribution and create two datasets, one with small values of $X_{2i}$ and one with large values.
4. Run two regression models, $RSS_2/RSS_1 \sim F(\frac{n-c}{2} - K - 1,\ \frac{n-c}{2} - K - 1)$. If $F_{\text{computed}} > F_{\text{critical}}$, then reject the null.

### 8.9.2 Breusch-Pagan and Cook-Weisberg Tests

There are important limitations of the GQ test. First, we must know something about the nature of heteroskedasticity. This is often doubtful, particularly in exploratory work.

Second, it's not **powerful** in small samples, which can be read: It's difficult to reject a false null because of the lack of power, increasing Type II error.

Third, remember, the null is homoskedastic variance, but if we retain the null, does that mean it is true? This leads to an alternative, the BP test. The logic of this test is somewhat different. It's nearly identical to another test, the Cook-Weisberg test. Here, we'll examine if particular variables are systematically related to the variance.

BP/CW both use the residuals from a fitted regression model as a proxy for the variance. The residuals are regressed on the independent variables and the null is $H_0 : \beta_2 = \beta_3 = \cdots = \beta_k = 0$.

Let's assume that heteroskedasticity is expected to be a function of several variables in our model, so:

$$U_i = \beta_1 + \beta_2 X_{2i} + \beta_3 X_{3i} + \omega_i$$

### 8.9.3 BP Test in Five Steps

1. Construct the null hypothesis: $H_0 : \beta_2 = \beta_3 = \cdots = \beta_k = 0$
2. Estimate a regression model and obtain residuals $e_1, \ldots, e_n$
3. Define $\tilde{\sigma}^2 = \frac{\sum e_i^2}{n}$. We use this to standardize/normalize the residuals. One thing to note: this is the variance for the maximum likelihood estimator.
4. Calculate $U_i = \frac{e_i^2}{\tilde{\sigma}^2}$
5. Estimate $U_i = \beta_1 + \beta_2 X_{2i} + \beta_3 X_{3i} + \omega_i$ and obtain the regression sum of squares. Breusch and Pagan show that RegSS/2 $\sim \chi^2_K$. If $\chi^2_{\text{computed}} > \chi^2_{\text{critical}}$, reject the null.

### 8.9.4 Cook-Weisberg (1983) Test

The test follows an almost identical set of steps, but instead of regressing $U_i$ on all the predictors, regress $U_i$ on $\widehat{Y}_i$:

$$U_i = \beta_1 + \beta_2 \widehat{Y}_i + \omega_i$$

Still use the formula: RegSS/2, which is distributed $\chi^2_1$. If $\chi^2_{\text{computed}} > \chi^2_{\text{critical}}$, reject the null.

**Comparing BP and CW**

These are useful tests. One doesn't need to know as much about the nature of heteroskedasticity. CW is a one-degree-of-freedom test, so is somewhat more powerful. It does not tell us which variable(s) lead to non-constant variance. Be careful when applying BP/CW in small samples.

### 8.9.5 White's Test

Recall that heteroskedasticity means $E(\epsilon\epsilon^T) \neq \sigma^2\mathbf{I}$. White [1980] proposes estimating $E(\epsilon\epsilon^T)$ directly, $\hat{\Sigma}$. This doesn't require knowledge of $\Omega$ in the variance equation: $\sigma_i^2 = \sigma^2\omega_i$.

White's test is no more than a modification/extension of the BP test.

**White's Test in Four Steps:**

1. Construct the null hypothesis: no heteroskedasticity.
2. Estimate a regression model and obtain residuals $e_1, \ldots, e_n$, then square the residuals $e_i^2$.
3. Regress $e_i^2$ on all variables, their squares, and their products:

$$e_i^2 = \beta_1 + \beta_2 X_{2i} + \beta_3 X_{3i} + \beta_4 X_{2i}^2 + \beta_5 X_{3i}^2 + \beta_6 X_{2i} X_{3i} + \omega_i$$

4. By this logic, there will be $P = [K(K+3)/2] + 1$ parameters in this regression equation, where $K$ is the number of variables (the 1 accounts for the intercept). Then, obtain $R^2$ from this model.

White [1980] shows $n \times R^2 \sim \chi_P^2$. So, compare $\chi^2_{\text{computed}}$ to $\chi^2_{\text{critical}}$ with $P$ degrees of freedom. If the computed value is larger, then reject the null. Its strength is also a weakness – it doesn't require a priori knowledge about the nature of heteroskedasticity.

**My Practical Advice**

Arguably the most common solutions to heteroskedasticity are WLS and White's (robust) standard errors. It is recommended to attempt multiple approaches and examine if they converge on the same solution.

## 8.10 Applied Analysis: Arizona Precinct Data

Now let's put everything together with the Arizona precinct data we introduced at the beginning of the chapter. We follow the diagnostic workflow recommended by Fox [2015] – estimate the model, examine residual plots, run formal tests, and apply corrections. The `car` package [Fox, 2015] provides convenient functions that integrate these steps.

### 8.10.1 Estimation and Rescaling

We first fit the model with the raw predictors, then rescale to make the coefficients comparable.

The coefficient for income is tiny (on the order of $10^{-6}$) because these variables are on vastly different scales. Median income ranges in the tens of thousands, while the Gini index sits between 0 and 1. We fix this by rescaling each predictor to a 0–1 range and mean-centering.

Now the coefficients are directly comparable – each represents the effect of moving across the full range of that predictor (0 to 1), centered at zero.

### 8.10.2 Predicted Margins by Latino Population

What does the model actually predict? Figure 8.1 shows predicted Trump–Harris margins at incremental levels of percent Latino, holding all other predictors at their means. Each line represents a different "slice" of the Latino population distribution (10th, 25th, 50th, 75th, and 90th percentiles of the raw `pct_latino` variable), translated back to the scaled predictor.

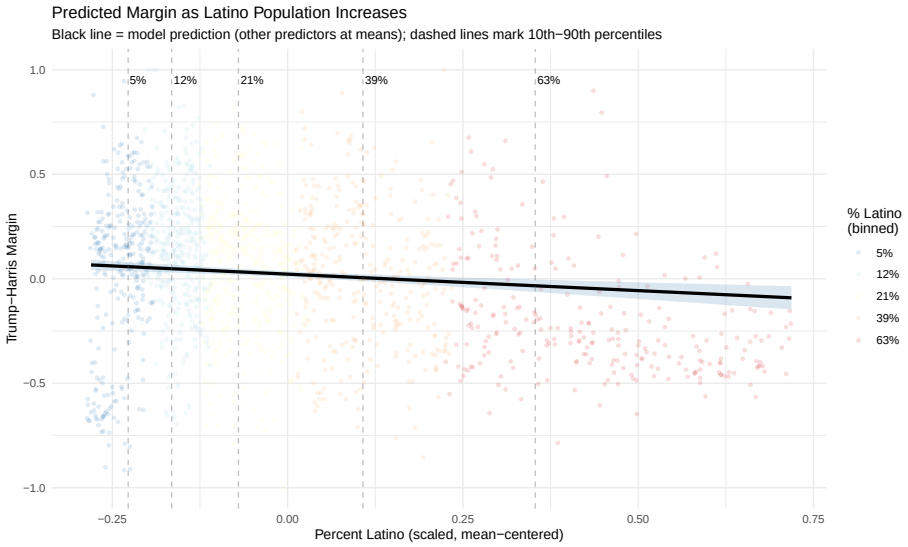



Figure 8.1: Predicted Trump–Harris margin across the range of percent Latino. Jittered observed data in the background. Lines show predictions at five representative levels of percent Latino, with other predictors at their means.

Notice how the **spread** of the jittered points changes across the range – precincts with very low or very high Latino percentages show different amounts of variability in the margin. This is exactly the visual signature of heteroskedasticity that we'll test formally below.

Table 8.3

```
fit_az_raw <- lm(trump_harris_margin ~ tract_acs_median_household_income +
                   pct_latino + tract_acs_median_age + tract_acs_gini_index,
                 data = precinct_voter_summary)
summary(fit_az_raw)
```

```
Call:
lm(formula = trump_harris_margin ~ tract_acs_median_household_income +
    pct_latino + tract_acs_median_age + tract_acs_gini_index,
    data = precinct_voter_summary)

Residuals:
    Min      1Q  Median      3Q     Max
-1.0910 -0.1948 -0.0197  0.1774  1.1668

Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                       1.567e-02  6.626e-
02   0.236    0.813
tract_acs_median_household_income  1.213e-06  2.187e-
07   5.545 3.40e-08 ***
pct_latino                        -1.575e-03  3.810e-04  -
4.134 3.75e-05 ***
tract_acs_median_age               1.048e-02  6.768e-
04  15.490  < 2e-16 ***
tract_acs_gini_index              -1.199e+00  1.115e-01 -
10.756  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2896 on 1674 degrees of freedom
  (9 observations deleted due to missingness)
Multiple R-squared:  0.2631,    Adjusted R-squared:  0.2613
F-statistic: 149.4 on 4 and 1674 DF,  p-value: < 2.2e-16
```

Table 8.4

```
# Rescale to 0-1 range, then mean-center
rescale01 <- function(x) {
  r <- (x - min(x, na.rm = TRUE)) / (max(x, na.rm = TRUE) - min(x, na.rm = TRUE))
  r - mean(r, na.rm = TRUE)
}

precinct_voter_summary <- precinct_voter_summary |>
  dplyr::mutate(
    income_sc = rescale01(tract_acs_median_household_income),
    latino_sc = rescale01(pct_latino),
    age_sc    = rescale01(tract_acs_median_age),
    gini_sc   = rescale01(tract_acs_gini_index)
  )

fit_az <- lm(trump_harris_margin ~ income_sc + latino_sc + age_sc + gini_sc,
             data = precinct_voter_summary)
summary(fit_az)
```

```
Call:
lm(formula = trump_harris_margin ~ income_sc + latino_sc + age_sc +
    gini_sc, data = precinct_voter_summary)

Residuals:
    Min      1Q  Median      3Q     Max
-1.0910 -0.1948 -0.0197  0.1774  1.1668

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.022187   0.007067   3.139  0.00172 **
income_sc    0.300229   0.054141   5.545 3.40e-08 ***
latino_sc   -0.157496   0.038101  -4.134 3.75e-05 ***
age_sc       0.622738   0.040203  15.490  < 2e-16 ***
gini_sc     -0.567660   0.052774 -10.756  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2896 on 1674 degrees of freedom
  (9 observations deleted due to missingness)
Multiple R-squared:  0.2631,    Adjusted R-squared:  0.2613
F-statistic: 149.4 on 4 and 1674 DF,  p-value: < 2.2e-16
```

### 8.10.3 Visual Diagnostics

The first step in detecting heteroskedasticity should be visual [Fox, 2015, Ch. 12] – arguably the first step should always be visual. Figure 8.2 plots the fitted values against the residuals. Under homoskedasticit this should look like a random cloud shape with constant spread. It should look like there is **no relationship between the fitted values and the residuals**.

Any systematic pattern – a funnel, a megaphone, a triangle shape – suggests otherwise.

```
plot_df <- data.frame(
  fitted = fitted(fit_az),
  residuals = residuals(fit_az)
)

ggplot(plot_df, aes(x = fitted, y = residuals)) +
  geom_point(alpha = 0.3, size = 1.5, color = "gray40") +
  geom_hline(yintercept = 0, color = "red", linewidth = 0.8) +
  geom_smooth(method = "loess", se = FALSE, color = "steelblue", linewidth = 1) +
  labs(x = expression(hat(Y) ~ "Fitted Values"),
       y = "Residuals",
       title = "Residuals v Fitted Values") +
  theme_minimal()
```

```
`geom_smooth()` using formula = 'y ~ x'
```

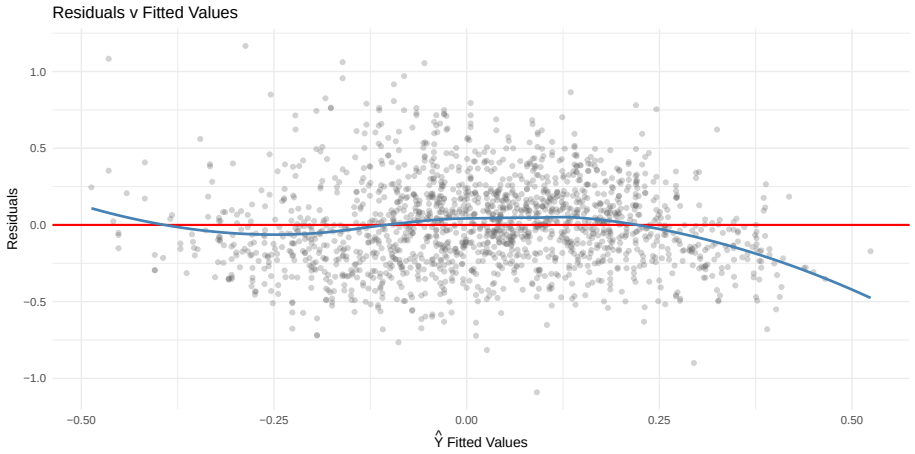



Figure 8.2: Residuals vs. fitted values for the Arizona precinct model. Notice the spread of residuals changes across the range of fitted values.

We can also check whether specific predictors drive non-constant variance. Figure 8.3 plots residuals against each predictor.

```
model_data <- fit_az$model
model_data$resid_sq <- residuals(fit_az)^2

p1 <- ggplot(model_data, aes(x = income_sc, y = resid_sq)) +
  geom_point(alpha = 0.3, size = 1) +
  geom_smooth(method = "loess", se = FALSE, color = "steelblue") +
  labs(x = "Median Household Income (scaled)", y = expression(e[i]^2)) +
  theme_minimal()

p2 <- ggplot(model_data, aes(x = latino_sc, y = resid_sq)) +
  geom_point(alpha = 0.3, size = 1) +
  geom_smooth(method = "loess", se = FALSE, color = "steelblue") +
  labs(x = "Latino Population (scaled)", y = expression(e[i]^2)) +
  theme_minimal()

p3 <- ggplot(model_data, aes(x = age_sc, y = resid_sq)) +
  geom_point(alpha = 0.3, size = 1) +
  geom_smooth(method = "loess", se = FALSE, color = "steelblue") +
  labs(x = "Median Age (scaled)", y = expression(e[i]^2)) +
  theme_minimal()

p4 <- ggplot(model_data, aes(x = gini_sc, y = resid_sq)) +
  geom_point(alpha = 0.3, size = 1) +
  geom_smooth(method = "loess", se = FALSE, color = "steelblue") +
  labs(x = "Gini Index (scaled)", y = expression(e[i]^2)) +
  theme_minimal()

gridExtra::grid.arrange(p1, p2, p3, p4, ncol = 2)
```

```
`geom_smooth()` using formula = 'y ~ x'
`geom_smooth()` using formula = 'y ~ x'
`geom_smooth()` using formula = 'y ~ x'
`geom_smooth()` using formula = 'y ~ x'
```

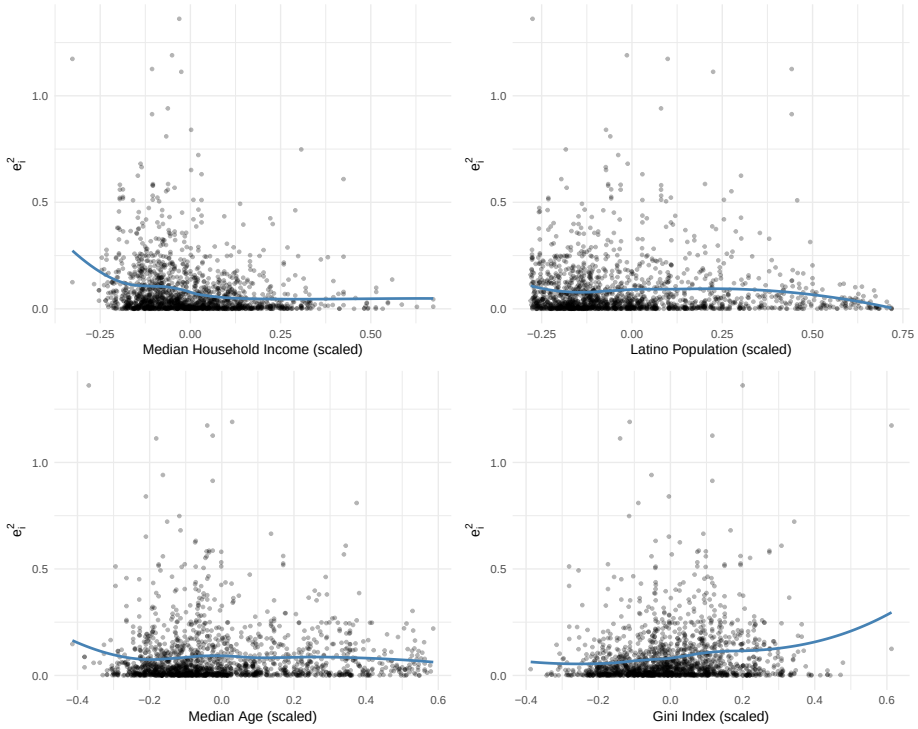



Figure 8.3: Squared residuals plotted against each predictor. Systematic trends indicate heteroskedasticity driven by that variable.

The `car` package provides two additional diagnostic tools: `residualPlots()` plots residuals against each predictor and the fitted values. `spreadLevelPlot()` plots $\log(|e_i|)$ against $\log(\hat{y}_i)$ – if the slope is non-zero, the variance depends on the level of the response [Fox, 2015, Ch. 12].

```
residualPlots(fit_az, tests = TRUE)
```

```
           Test stat Pr(>|Test stat|)
income_sc    -3.2187         0.001312 **
latino_sc    -7.5168        9.099e-14 ***
age_sc       -5.0448        5.033e-07 ***
gini_sc       1.5340         0.125215
Tukey test   -7.2550        4.017e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

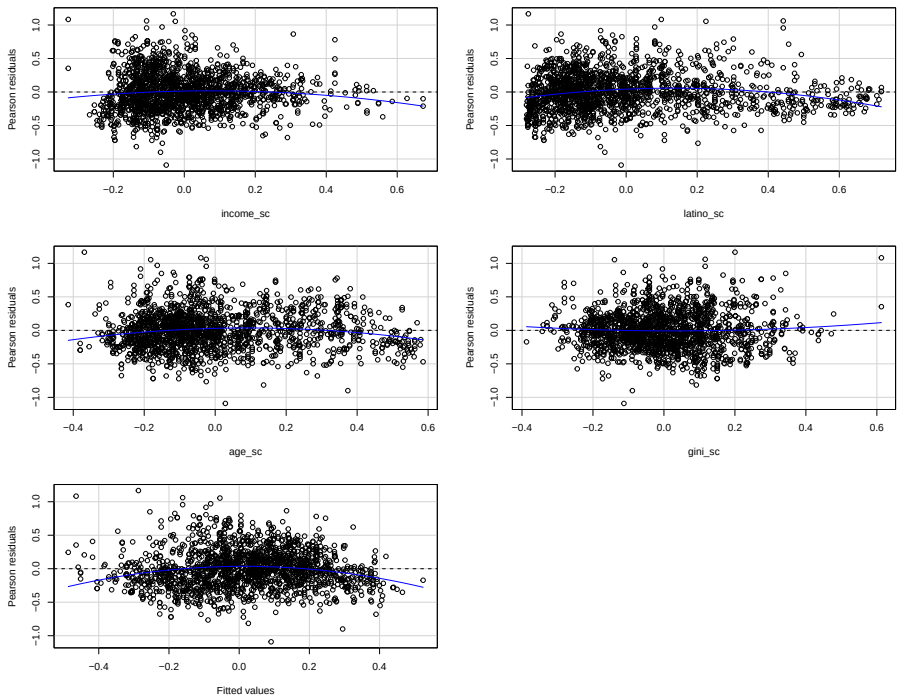



Figure 8.4: Residual plots from car::residualPlots(). Each panel shows residuals against a predictor, with a fitted curve testing for non-linearity.

```
=== Cook-Weisberg Score Test (from car::ncvTest) ===

Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 21.49103, Df = 1, p = 3.5549e-06

=== ncvTest against specific predictor (income) ===

Non-constant Variance Score Test
Variance formula: ~ income_sc
Chisquare = 58.54519, Df = 1, p = 1.9867e-14

=== ncvTest against specific predictor (latino) ===

Non-constant Variance Score Test
Variance formula: ~ latino_sc
Chisquare = 2.845904, Df = 1, p = 0.091607
```

The `ncvTest()` function from `car` implements the Cook-Weisberg score test directly. A significant result rejects the null of constant variance. Testing against specific predictors helps identify *which* variable drives the heteroskedasticity.

### 8.10.4 Formal Tests

Now let's apply the formal tests discussed above. We use `lmtest::bptest()` for the Breusch-Pagan test and construct White's test manually.

### 8.10.5 Robust Standard Errors

Given heteroskedasticity, we can correct the standard errors using White [1980]'s HC estimator [Fox, 2015, Ch. 12]. The `sandwich` package provides several variants (HC0 through HC5). Compare the OLS standard errors to the heteroskedasticity-consistent (HC) standard errors:

**Interpreting the Comparison**

If the robust SEs differ substantially from the OLS SEs, that's evidence of heteroskedasticity affecting inference. Coefficients that were significant under OLS may lose significance with robust SEs (or vice versa). The coefficient *estimates* don't change – only the standard errors and therefore the t-statistics and p-values.

**Recap**

We've now developed a full workflow for diagnosing and correcting heteroskedasticity:

1. Fit the model and examine residual plots for visual signs of non-constant variance.
2. Run formal tests (BP, CW, White) to statistically assess heteroskedasticity and identify potential drivers.
3. If heteroskedasticity is present, apply corrections such as robust standard errors or WLS, and compare results to OLS.
4. We've also directly seen how heteroskedasticity can distort inference by mis-estimating the standard errors. And also, notice that the point estimates, the slopes don't change.

### 8.10.6 Back to Precincts

Figure 8.5 shows the geographic distribution of residuals. If heteroskedasticity has a spatial pattern – for instance, if residuals are larger in rural areas with fewer precincts – this can help diagnose the source.

### 8.10.7 WLS Correction

If we suspect the variance scales with a predictor – say, precincts with larger Latino populations show more variability in margins – we can apply WLS. Here we weight by the inverse of the estimated variance function:

Table 8.5

```
library(lmtest)

# Breusch-Pagan Test
cat("=== Breusch-Pagan Test ===\n")
```

```
=== Breusch-Pagan Test ===
```

```
bptest(fit_az)
```

```
    studentized Breusch-Pagan test

data:  fit_az
BP = 76.748, df = 4, p-value = 8.504e-16
```

```
# Cook-Weisberg (studentized BP against fitted values)
cat("\n=== Cook-Weisberg Test (against fitted values) ===\n")
```

```
=== Cook-Weisberg Test (against fitted values) ===
```

```
aux_data <- fit_az$model
aux_data$yhat <- fitted(fit_az)
bptest(fit_az, ~ yhat, data = aux_data)
```

```
    studentized Breusch-Pagan test

data:  fit_az
BP = 17.057, df = 1, p-value = 3.627e-05
```

```
# White's Test (BP against all predictors, squares, and cross-products)
cat("\n=== White's Test ===\n")
```

```
=== White's Test ===
```

```
aux_data$inc2  <- aux_data$income_sc^2
aux_data$lat2  <- aux_data$latino_sc^2
aux_data$age2  <- aux_data$age_sc^2
aux_data$gini2 <- aux_data$gini_sc^2

bptest(fit_az, ~ income_sc + latino_sc + age_sc + gini_sc +
                  inc2 + lat2 + age2 + gini2 +
                  income_sc:latino_sc +
                  income_sc:age_sc +
                  income_sc:gini_sc +
                  latino_sc:age_sc +
                  latino_sc:gini_sc +
                  age_sc:gini_sc,
       data = aux_data)
```

```
    studentized Breusch-Pagan test

data:  fit_az
BP = 114.25, df = 14, p-value < 2.2e-16
```

Table 8.6

```
library(sandwich)
library(lmtest)

cat("=== OLS Standard Errors ===\n")
```

```
=== OLS Standard Errors ===
```

```
coeftest(fit_az)
```

```
t test of coefficients:

              Estimate Std. Error  t value  Pr(>|t|)
(Intercept)  0.0221865  0.0070673   3.1393  0.001723 **
income_sc    0.3002286  0.0541406   5.5454 3.403e-08 ***
latino_sc   -0.1574963  0.0381011  -4.1336 3.747e-05 ***
age_sc       0.6227382  0.0402027  15.4900 < 2.2e-16 ***
gini_sc     -0.5676602  0.0527741 -10.7564 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
cat("\n=== HC1 (White) Robust Standard Errors ===\n")
```

```
=== HC1 (White) Robust Standard Errors ===
```

```
coeftest(fit_az, vcov = vcovHC(fit_az, type = "HC1"))
```

```
t test of coefficients:

              Estimate Std. Error  t value  Pr(>|t|)
(Intercept)  0.0221865  0.0070644   3.1406  0.001716 **
income_sc    0.3002286  0.0527272   5.6940 1.463e-08 ***
latino_sc   -0.1574963  0.0389254  -4.0461 5.445e-05 ***
age_sc       0.6227382  0.0426856  14.5890 < 2.2e-16 ***
gini_sc     -0.5676602  0.0527222 -10.7670 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

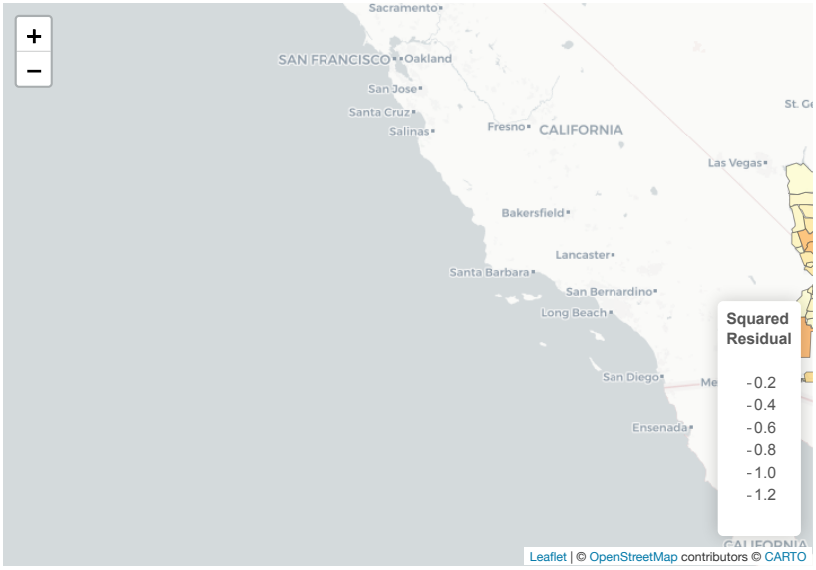



Figure 8.5: Arizona precincts colored by squared residuals from the margin model. Larger values indicate precincts where the model fits poorly.

Table 8.7

```
# Build WLS data from the fitted model
wls_data <- fit_az$model
wls_data$resid_sq <- residuals(fit_az)^2

# Estimate variance function: regress squared residuals on predictors
var_model <- lm(resid_sq ~ income_sc + latino_sc + age_sc + gini_sc, data = wls_data)
wls_data$sigma_hat <- sqrt(abs(fitted(var_model)))

# The weight is 1/sigma_hat -- matching rho_ii = 1/sqrt(omega_ii).
wls_data$w <- 1 / wls_data$sigma_hat

# WLS model via glm
fit_wls <- glm(trump_harris_margin ~ income_sc + latino_sc + age_sc + gini_sc,
               data = wls_data,
               weights = w,
               family = gaussian())

cat("=== WLS Estimates ===\n")
```

```
=== WLS Estimates ===
```

```
summary(fit_wls)
```

```
Call:
glm(formula = trump_harris_margin ~ income_sc + latino_sc + age_sc +
    gini_sc, family = gaussian(), data = wls_data, weights = w)

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.020781   0.006979   2.977  0.00295 **
income_sc    0.197078   0.045507   4.331 1.57e-05 ***
latino_sc   -0.233373   0.037371  -6.245 5.37e-10 ***
age_sc       0.545278   0.040833  13.354  < 2e-16 ***
gini_sc     -0.565155   0.050654 -11.157  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.2879098)

    Null deviance: 641.90  on 1678  degrees of freedom
Residual deviance: 481.96  on 1674  degrees of freedom
AIC: 538.05

Number of Fisher Scoring iterations: 2
```

Table 8.8

```
cat("=== Comparison of Standard Errors ===\n\n")
```

```
=== Comparison of Standard Errors ===
```

```
ols_se <- sqrt(diag(vcov(fit_az)))
robust_se <- sqrt(diag(vcovHC(fit_az, type = "HC1")))
wls_se <- sqrt(diag(vcov(fit_wls)))

comparison <- data.frame(
  OLS = round(ols_se, 6),
  Robust_HC1 = round(robust_se, 6),
  WLS = round(wls_se, 6)
)
comparison
```

```
                OLS Robust_HC1      WLS
(Intercept) 0.007067   0.007064 0.006979
income_sc   0.054141   0.052727 0.045507
latino_sc   0.038101   0.038925 0.037371
age_sc      0.040203   0.042686 0.040833
gini_sc     0.052774   0.052722 0.050654
```

**What to Look For**

Compare the three columns. Where the robust and WLS standard errors diverge from OLS, that's how heteroskedasticity distorts inference.

**Clarification and Tools**

R offers several functions that can fit weighted models.
`lm(..., weights = w)` fits weighted least squares directly. The `weights` argument specifies $w_i = 1/\text{var}(\epsilon_i)$. This is the simplest approach and works well when you can supply the weights yourself.
`glm(..., weights = w, family = gaussian())` does the same thing when the family is Gaussian with an identity link – this is the default. We'll use this approach in POL 683. The coefficient estimates, standard errors, and residuals are identical to `lm()` with the same weights. This approach is a little more flexible.

## 8.11 Concluding Remarks

Heteroskedasticity is a commonly violated assumption of the Gauss-Markov assumptions.

### 8.11.1 Steps to Address Heteroskedasticity

1. **Understanding the problem.** When $E(\epsilon\epsilon^T) = \sigma^2\Omega$ with $\Omega \neq \mathbf{I}$, OLS remains unbiased but is no longer efficient – **the standard errors are wrong**. Hypothesis tests and confidence intervals are based on the usual $\sigma^2(\mathbf{X}^T\mathbf{X})^{-1}$ which is wrong.
2. **Detection.** Visual diagnostics (residual plots, spread-level plots) provide the first indication. Formal tests – Breusch-Pagan, Cook-Weisberg, and White's test – offer statistical confirmation and can help identify which predictors drive the non-constant variance.
3. **Correction.** Two main strategies:
   - **Robust standard errors** (HC1, HC3) correct the inference without changing the coefficient estimates. T
   - **Weighted Least Squares** transforms the data so that OLS on the transformed data is BLUE. WLS is more efficient when the variance structure is correctly specified, but it requires knowing (or estimating) the weights.
4. **Practical judgment.** The choice between robust SEs and WLS depends on how much you know about the variance structure.

# Chapter 9

# Multicollinearity

## 9.1 Setup

We've spent a good deal of time discussing the properties of the OLS estimator – unbiasedness, efficiency, and so forth. We've also examined what happens when assumptions like homoskedasticity are violated. Now we turn to a different kind of problem, one that doesn't technically violate any Gauss-Markov assumption, but can adversely impact your results nonetheless: multicollinearity.

## 9.2 Data Problem, or Estimator Problem?

We can split multicollinearity into two categories: perfect and imperfect. **Perfect multicollinearity** is a *data problem*. It means the data simply do not contain enough information to estimate the model. **Imperfect multicollinearity** is an estimator problem. The data contain enough information, but the OLS estimator is not doing a good job of extracting it.

## 9.3 Perfect Collinearity

With perfect collinearity, the OLS estimators are not defined, there is no OLS solution. Consider the model,

$$y_i = \beta_1 + \beta_2 X_1 + \beta_3 X_2 + \epsilon_i$$

When $r_{X_1 X_2} = 1$, the denominator for $\hat{\beta}_2$ and $\hat{\beta}_3$ is zero, and $\mathbf{X^T X}$ is singular. You literally cannot invert the matrix – the estimator $(\mathbf{X^T X})^{-1}\mathbf{X^T y}$ does not exist.

Why? Think about it intuitively. If $X_1$ and $X_2$ are perfectly correlated, there is no way to tell them apart. The regression cannot "decide" how much of the variation in $y$ to attribute to $X_1$ versus $X_2$. The variables carry identical information.

More precisely, we should use the term "multicollinearity." If one variable is a perfect linear combination of $k$ other variables in our model, the same problems exists:

$$X_j = C_1 d_1 + \cdots + C_k d_k$$

### 9.3.1 Dummy Variables and Multicollinearity

If any variable is determined by some combination of $X_k$ (i.e., $C \neq 0$), there will exist **perfect** multicollinearity. This problem is encountered in different ways, but let's consider dummy variables.

Recall from our earlier discussion of categorical predictors. With $k$ categories, we include $k-1$ dummies; the omitted group serves as the **reference category**. The reason we omit one is precisely multicollinearity. Let's consider party identification, coded as Republican, Independent, and Democrat. We create dummy variables:

- $D_{\text{Rep}} = 1$ if Republican, 0 otherwise
- $D_{\text{Dem}} = 1$ if Democrat, 0 otherwise
- $D_{\text{Ind}} = 1$ if Independent, 0 otherwise

Notice that $D_{\text{Rep}} + D_{\text{Dem}} + D_{\text{Ind}} = 1$ for every observation – because every person is exactly one of these three things. But the intercept column in $\mathbf{X}$ is also a column of 1s. So the three dummy columns sum to produce the intercept column. That's a perfect linear dependency among the columns of $\mathbf{X}$, which means $\mathbf{X^T X}$ is singular and we cannot estimate the model. This is the "dummy variable trap."

Why? The model with all $k$ dummies and an intercept is:

$$Y_i = \alpha + \gamma_{\text{Rep}} D_{\text{Rep}} + \gamma_{\text{Dem}} D_{\text{Dem}} + \gamma_{\text{Ind}} D_{\text{Ind}} + \beta X + e_i$$

But since $D_{\text{Ind}} = 1 - D_{\text{Rep}} - D_{\text{Dem}}$, the Independent dummy is a perfect linear function of the other two dummies and the intercept. There is no unique solution.

What happens in `R` when this occurs? It will just drop one variable to estimate the model. When we drop one category (say, Independents), the intercept now represents the baseline for that group, and the remaining dummy coefficients represent differences *from* that baseline. An Independent is simply a case when both $D_{\text{Rep}} = 0$ and $D_{\text{Dem}} = 0$. The choice of reference category is arbitrary in terms of $\hat{Y}_i$ – you'll get the same predicted values regardless – but remember that it will change the interpretation of the dummy coefficients.

```
set.seed(42)
n <- 100
party <- sample(c("Republican", "Independent", "Democrat"), n, replace = TRUE)
auth <- runif(n, 0, 1)
y_trust <- 0.5 + 0.3 * (party == "Republican") - 0.2 * (party == "Democrat") +
    0.4 * auth + rnorm(n, 0, 0.3)

d_rep <- as.numeric(party == "Republican")
d_dem <- as.numeric(party == "Democrat")
d_ind <- as.numeric(party == "Independent")

# Include all 3 dummies -- R drops one
cat("All 3 dummies (R drops one):\n")
```

```
All 3 dummies (R drops one):
```

```
coef(lm(y_trust ~ d_rep + d_dem + d_ind + auth))
```

```
(Intercept)       d_rep       d_dem       d_ind        auth
  0.3916643   0.3492268  -0.2866816          NA   0.5729838
```

```
# The correct specification: k-1 dummies
cat("\nk-1 dummies (Independents as reference):\n")
```

```
k-1 dummies (Independents as reference):
```

```
coef(lm(y_trust ~ d_rep + d_dem + auth))
```

```
(Intercept)       d_rep       d_dem        auth
  0.3916643   0.3492268  -0.2866816   0.5729838
```

Notice that `R` sets the coefficient on `d_ind` to `NA` – it detects the perfect linear dependency and removes a redundant variable.

And once again, the two specifications produce identical predicted values. It's entirely avoidable once you understand that one variable is linearly dependent on the others, specifically that $\sum_{j=1}^{k} D_j = 1$.

```
# Create data with a perfect linear dependency
set.seed(42)
n <- 100
x1 <- rnorm(n, 10, 2)
x2 <- 2 * x1 + 5 # x2 is a perfect linear function of x1
y <- 3 + 1.5 * x1 + rnorm(n)

# Try to fit the model
model_perfect <- lm(y ~ x1 + x2)
summary(model_perfect)
```

```
Call:
lm(formula = y ~ x1 + x2)

Residuals:
     Min       1Q   Median       3Q      Max
-1.88842 -0.50664  0.01225  0.54106  2.86240

Coefficients: (1 not defined because of singularities)
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.77584    0.45043   6.163 1.59e-08 ***
x1           1.51358    0.04383  34.531  < 2e-16 ***
x2                NA         NA      NA       NA
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9083 on 98 degrees of freedom
Multiple R-squared:  0.9241,    Adjusted R-squared:  0.9233
F-statistic:  1192 on 1 and 98 DF,  p-value: < 2.2e-16
```

### 9.3.2 The Mathematical Problem

Let's connect this back to the formulas we derived earlier; the model coefficients:

$$\mathbf{b} = (\mathbf{X^T X})^{-1}\mathbf{X^T y}$$

**This requires $\mathbf{X^T X}$ to be invertible.** And recall – a matrix is invertible only if its determinant is nonzero. When one column of $\mathbf{X}$ is a linear combination of the others, $\det(\mathbf{X^T X}) = 0$, the matrix is **singular**, and no inverse exists.

## 9.4 Imperfect Multicollinearity

More frequently, we are in the situation where $0 < r_{X_1 X_2} < 1$. This is now a **data problem**, and the OLS estimator is still BLUE. Imperfect collinearity is not a violation of the Gauss-Markov assumptions. The OLS estimator is still unbiased, and the variance is still correctly estimated.

So why should we care? The problem is that the variance of the OLS estimator can become very large, which means our estimates are very imprecise.

The data provide enough information to estimate the model, but the OLS estimator is not able to extract *precise* estimates.

This type of collinearity is also incredibly common. Variables are correlated – sometimes strongly – but not perfectly. Education and income are correlated. Age and experience are correlated. In political science, partisanship and ideology are correlated. The question is: how much correlation is too much?

We discussed the variance of the estimate in Part I. Another way to write the of the OLS estimator is given by,

$$\text{var}(b_j) = \frac{1}{1 - R_j^2} \times \frac{\sigma^2}{\sum x_i^2}$$

where $R_j^2$ is the $R^2$ from regressing $X_j$ on all the other independent variables in the model. This is important – it's not the $R^2$ from the main regression. It's the $R^2$ from the *auxiliary* regression of one predictor on all the others. The term,

$$\frac{1}{1 - R_j^2} = \text{VIF}$$

is called the Variance Inflation Factor. Or equivalently, $\sqrt{\text{VIF}}$ tells us how much the standard error of $b_j$ is inflated relative to the case with no collinearity.

As the VIF increases, $\text{var}(b_j)$ will increase. But the OLS estimator is still BLUE. The problem is not whether one can obtain an unbiased estimate – we definitely can. We can also obtain the "correct" variance – but it may be quite large. This is the key point that practitioners often miss. Multicollinearity is not a *bias* problem. It is a *precision* problem. Your estimates are right on average, but any single estimate may be way off.

**OLS is still BLUE in the presence of imperfect multicollinearity. The estimator is unbiased. The variance is correctly estimated. The problem is that the variance itself is large. Don't confuse "large variance" with "biased estimates" – these are fundamentally differrent things. Think back to how we statistically demonstrated these concepts.**

Let's see how this works in practice.

- Let's start by simulating data, but let's simulate data such that the correlation between our predictor varies. Then let's estimate the model to see what happens to the standard errors as the correlation changes.
- We'll simulate data and estimate the models repeatedly, varying the correlation parameter.
- We'll then plot the standard errors as a function of the correlation between the predictors.
- We'll also plot the VIF, as well as the slope estimates.

Three things to notice.

First, the standard errors increase drastically as the correlation approaches 1. At $r = 0.9$, the standard errors are already more than double what they are at $r = 0$. At $r = 0.99$, they're rather large.

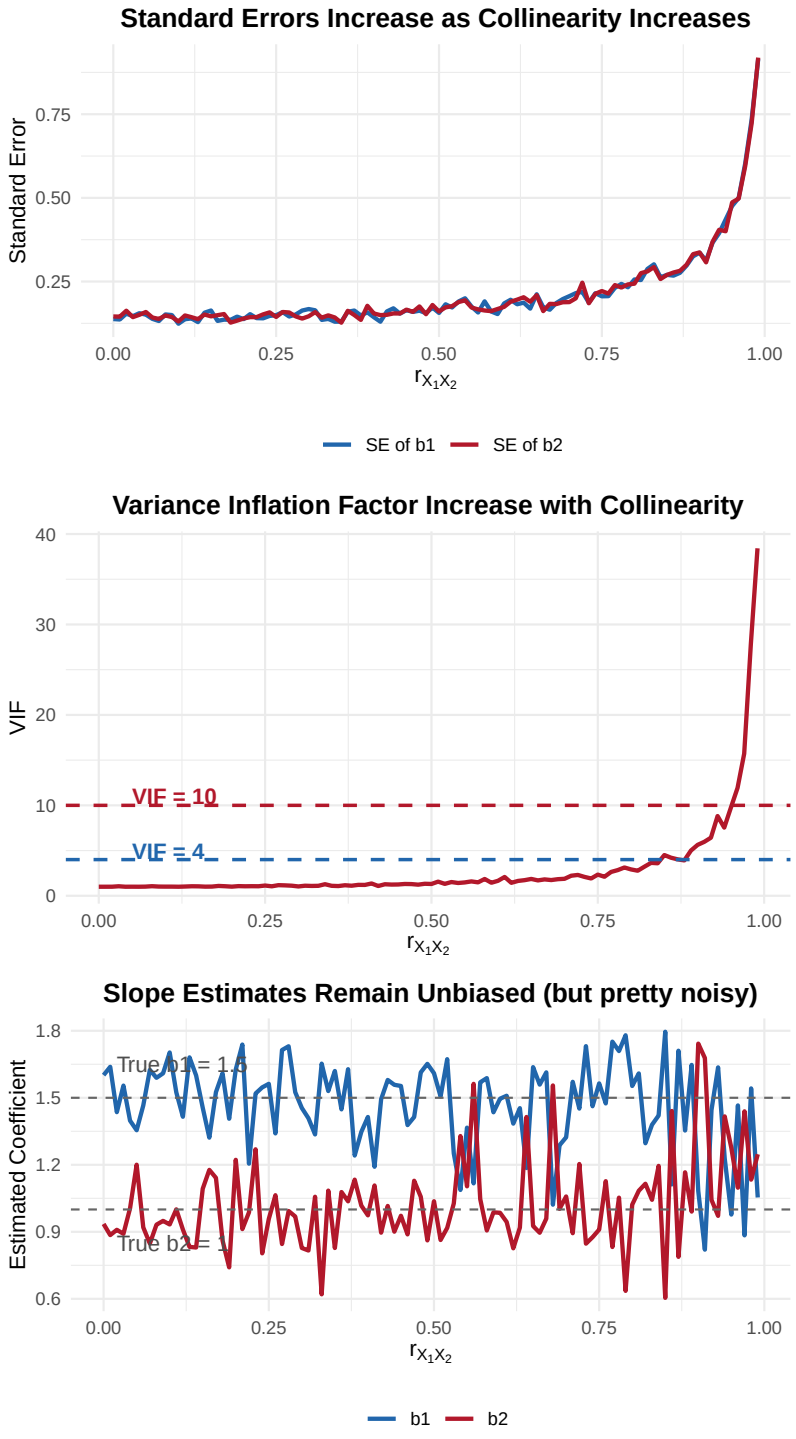



Figure 9.1: As collinearity increases: standard errors explode, VIF rises, but slopes remain unbiased

Second, the VIF tracks this exactly. It crosses the the threhold of 4 around $r \approx 0.87$, and 10 around $r \approx 0.95$.

Third – and this is the critical point – the slope estimates bounce around their true values at every level of correlation. They are still **unbiased**. The estimates get noisier as collinearity increases, but they are centered on the true value. Again, the estimates are BLUE but noisy.

## 9.5 Why This Happens: The "No Sniffles" Problem

Think about it this way. Suppose $X_1$ and $X_2$ are uncorrelated. When we add $X_2$ to the model, the variation in $X_1$ that "explains" $y$ is distinct from the variation in $X_2$ that explains $y$. Now suppose $X_1$ and $X_2$ are highly correlated. Most of the variation in $X_1$ is also present in $X_2$. The regression has to parse out each variable's uniqueness – how much each contributes to $y$.

Let's make this concrete with a fictious – and cautionary – example. A startup firm has just launched **"No Sniffles"** a homeopathic allergy drug. To build their case for efficacy, they analyze data from a large **non-randomized** survey of Arizona residents conducted during peak allergy season (March through June). Respondents report how many doses of "No Sniffles" they took, how many **pain relievers** (Ibuprofen, Advil, etc.) they took, and answer the question: *"How much was your headache reduced?"* on a 0–10 scale.

During allergy season in Arizona, pollen triggers sinus headaches. People who suffer from allergies tend to reach for *both* an allergy remedy *and* a pain reliever at the same time. The usage of "No Sniffles" and pain relievers are **correlated** – not because one causes the other, but because allergy season drives both. The startup's regression will struggle to isolate "No Sniffles"' unique effect on headache reduction from the effect of pain relievers.

Notice the structure: allergy season drives people to take *both* drugs, creating the correlation. The truth is that pain relievers genuinely and substantially reduce headaches ($\beta_2 = 1.8$), but "No Sniffles" – being homeopathic – has only a very modest placebo effect ($\beta_1 = 0.15$). Can the startup's regression tell these apart?

### 9.5.1 Simulating the Survey

Let's simulate 800 Arizona residents surveyed during March–June. The **true** model is:

$$\text{Headache Reduction}_i = 1.0 + 0.15 \times \text{NoSniffles}_i + 1.8 \times \text{PainRelievers}_i + \epsilon_i$$

"No Sniffles" barely helps (true $\beta_1 = 0.15$), but pain relievers strongly reduce

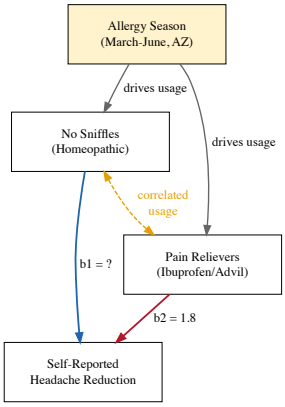



Figure 9.2: During Arizona allergy season, usage of 'No Sniffles' and pain relievers are correlated – creating collinearity that makes it hard to isolate each drug's effect

headaches (true $\beta_2 = 1.8$). The problem: during allergy season, people reach for both.

```
library(MASS)

set.seed(2024)
n <- 800

# True parameters
beta_0 <- 1.0
beta_nosniffles <- 0.15
beta_painrelief <- 1.8

doses <- MASS::mvrnorm(n,
    mu = c(2, 2.5),
    Sigma = matrix(c(1.0, 0.55, 0.55, 1.2), nrow = 2)
)
no_sniffles <- pmax(round(doses[, 1]), 0)
pain_relievers <- pmax(round(doses[, 2]), 0)

month <- sample(3:6, n, replace = TRUE)

headache_reduction <- beta_0 + beta_nosniffles * no_sniffles +
    beta_painrelief * pain_relievers + rnorm(n, 0, 1.0)
headache_reduction <- pmin(pmax(headache_reduction, 0), 10)

survey_data <- data.frame(
    no_sniffles, pain_relievers, headache_reduction,
    month = factor(month, labels = c("March", "April", "May", "June"))
)
```

### 9.5.2 Full Sample: Can We Tell the Drugs Apart?

With all 800 respondents, the correlation between "No Sniffles" and pain reliever usage is moderate. Let's see what the regression finds.

```
library(car)

cat(
    "Correlation between No Sniffles and Pain Relievers (full sample):",
    round(cor(survey_data$no_sniffles, survey_data$pain_relievers), 3), "\n\n"
)
```

```
Correlation between No Sniffles and Pain Relievers (full sample): 0.478
```

```
model_full <- lm(headache_reduction ~ no_sniffles + pain_relievers,
    data = survey_data
```

```
)
summary(model_full)
```

```
Call:
lm(formula = headache_reduction ~ no_sniffles + pain_relievers,
    data = survey_data)

Residuals:
     Min       1Q   Median       3Q      Max
-3.03170 -0.64028  0.00806  0.67851  3.08922

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     1.23469    0.09253  13.344   <2e-16 ***
no_sniffles     0.08968    0.03742   2.397   0.0168 *
pain_relievers  1.75255    0.03599  48.695   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9692 on 797 degrees of freedom
Multiple R-squared:  0.8018,    Adjusted R-squared:  0.8013
F-statistic:  1612 on 2 and 797 DF,  p-value: < 2.2e-16
```

```
cat("\nVIF:\n")
```

```
VIF:
```

```
vif(model_full)
```

```
   no_sniffles pain_relievers
      1.295352       1.295352
```

With moderate collinearity, we can *probably* tell the drugs apart – pain relievers show a strong effect, and "No Sniffles" shows a small one.

### 9.5.3 Subsetting: Increasing the Collinearity

Imagine different sampling strategies the startup might use. As we restrict the sample to respondents whose "No Sniffles" and pain reliever doses are more similar. There is a stronger relationship. What happens?

```
library(car)

max_diffs <- c(Inf, 3, 2, 1)
subset_labels <- c("Full Sample", "|Diff| <= 3", "|Diff| <= 2", "|Diff| <= 1")

results <- data.frame()
```

```
for (i in seq_along(max_diffs)) {
    d <- max_diffs[i]
    sub <- survey_data[abs(survey_data$no_sniffles -
        survey_data$pain_relievers) <= d, ]
    mod <- lm(headache_reduction ~ no_sniffles + pain_relievers, data = sub)
    v <- vif(mod)
    s <- summary(mod)

    results <- rbind(results, data.frame(
        Subset = subset_labels[i],
        N = nrow(sub),
        r_xy = round(cor(sub$no_sniffles, sub$pain_relievers), 3),
        b_nosniffles = round(coef(mod)["no_sniffles"], 3),
        se_nosniffles = round(s$coefficients["no_sniffles", 2], 3),
        b_painrelief = round(coef(mod)["pain_relievers"], 3),
        se_painrelief = round(s$coefficients["pain_relievers", 2], 3),
        VIF = round(v[1], 2)
    ))
}

knitr::kable(results,
    col.names = c(
        "Subset", "N", "r(NS, PR)", "b_NoSniffles", "SE_NoSniffles",
        "b_PainRelief", "SE_PainRelief", "VIF"
    )
)
```

Table 9.1: As we restrict to respondents with similar dosing patterns, collinearity increases and the regression can no longer distinguish the effects

| | Subset | N | r(NS, PR) | b_NoSniffles | SE_NoSniffles | b_PainRelief | SE_PainRelief | VIF |
|---|---|---|---|---|---|---|---|---|
| no_sniffles | Full Sample | 800 | 0.478 | 0.090 | 0.037 | 1.753 | 0.036 | 1.30 |
| no_sniffles1 | \|Diff\| <= 3 | 799 | 0.482 | 0.092 | 0.038 | 1.750 | 0.036 | 1.30 |
| no_sniffles2 | \|Diff\| <= 2 | 769 | 0.561 | 0.094 | 0.041 | 1.761 | 0.040 | 1.46 |
| no_sniffles3 | \|Diff\| <= 1 | 636 | 0.747 | 0.059 | 0.059 | 1.818 | 0.056 | 2.26 |

The startup's problem is now obvious. With the full sample and moderate collinearity, the regression *can* tell the drugs apart: pain relievers have a large, significant effect, while "No Sniffles" a more modest one.

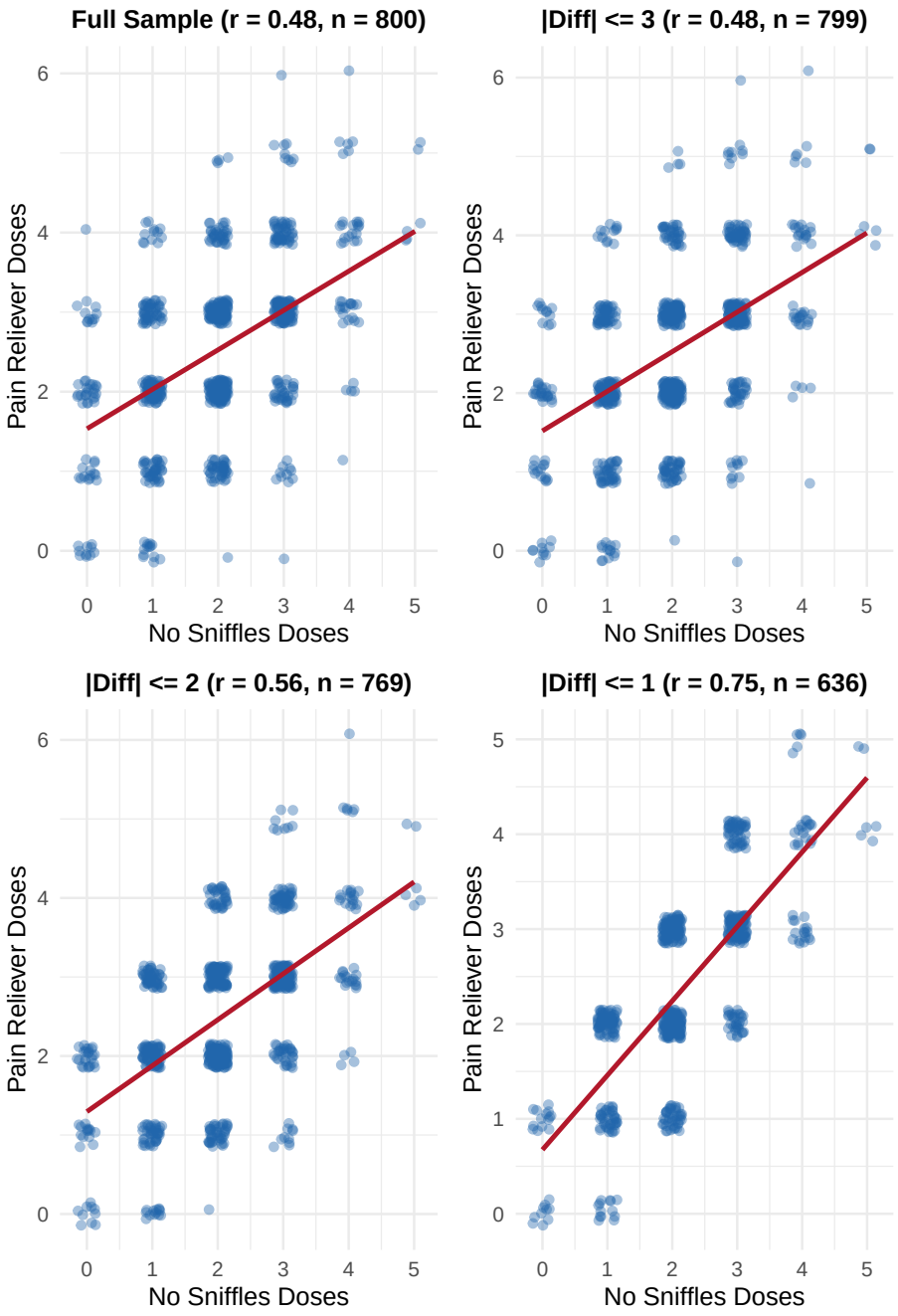



Figure 9.3: As we restrict to similar dosing patterns, the data cloud aligns along the diagonal – collinearity makes it impossible to separate the two effects

But if the correlation is higher– people who always take both together – the standard errors inflate, the VIF increases, and we can no longer distinguish a drug that genuinely works ($\beta_2 = 1.8$) from one that barely works ($\beta_1 = 0.15$).

### 9.5.4 Randomization

This is exactly the danger of non-randomized data with correlated treatments. A randomized trial would *independently* assign "No Sniffles" and pain reliever usage; this then break the correlation between right hand side variables. The observational survey during allergy season – where everyone reaches for both drugs at once – cannot do this.

## 9.6 Detecting Multicollinearity

When does this matter? What should we do?

- You obtain a large $R^2$ value, significant F values, but non-significant t-tests (and large confidence intervals). This is the classic indicator. The model as a whole is explaining variation in $y$, but you can't pin down a particular effect. Everything is non-significant. There is a divergence between the F-test – an indicator of fit – and the t-tests (the test of individual coefficients).
- To explore if this is problematic, we might first obtain $r_{xx}$, the correlation matrix of independent variables, and you observe an entry of 0.80 or greater. But, remember, this may not be immediately transparent from the table, if $X_k$ is a linear combination of some set of variables. A variable might not be highly correlated with any single other variable, but could be strongly predicted by a *combination* of variables.
- Examine the VIF. If VIF exceeds 4, or $\sqrt{\text{VIF}}$ exceeds 2, collinearity may be an issue. Some textbooks use 10 as the threshold. It's an indicator, a useful heuristic.

## 9.7 Common Mistakes

This is where students – and researchers – tend to go wrong. Let me walk through the mistakes I see most often.

### 9.7.1 Mistake 1: Dropping Variables to "Fix" Collinearity

The most common response to multicollinearity is to drop one of the correlated variables. Sometimes this is the right thing to do. Often it is not.

Suppose you're modeling Trump's vote margin in Arizona precincts as a function of median household income and median age. They're correlated in your data (older populations often live in areas with different income distributions), so the

standard errors are large. You drop median age. Now the standard errors on income are small and the coefficient is significant. Fixed it. Right?

If median age *actually affects* voting patterns, then dropping it introduces **omitted variable bias**. You've traded a precision problem for a bias problem. The coefficient on income now captures *both* the direct effect of income *and* the indirect effect through its correlation with age. Your estimate is biased – and a biased estimate with a small standard error can be worse than an unbiased estimate with a large one.

**Do not drop theoretically important variables just because they are correlated with other predictors. An unbiased estimate with a large standard error is preferable to a biased estimate with a small standard error. If you can't reject the null, that's informative – it means the data don't contain enough information to distinguish between the effects.**

### 9.7.2 Mistake 2: Concluding That the Variables "Don't Matter"

When multicollinearity inflates standard errors and you fail to reject $H_0 : \beta_j = 0$, it's tempting to conclude that the variables don't affect $y$. It now becomes an issue of statisticaly power – you simply don't have enough information to distinguish between the effects – but that doesn't mean those effects aren't there. Failing to reject a null does not mean the null is true.

Instead, it means you lack sufficient evidence to conclude that the variable has an effect, *given the other variables in the model*. With highly collinear predictors, the individual t-tests have low power; our ability to correctly reject a null hypothesis when it is false is diminished. You *should* expect non-significant results, even if the variables truly matter.

### 9.7.3 Mistake 3 Using Stepwise Regression

You'll occasionally hear about stepwise regression procedures, where analysts iterate through different specifications and settle on a model. Maybe then the researcher does some additional work to justify the final model. This is atheoretical and should be avoided. Recall that a "correct" regression model must be correctly specified – and that requires theory, not data-driven selection.

### 9.7.4 Mistake 4: Ignoring the Problem Entirely

On the other end, some researchers simply ignore multicollinearity and report their results as if nothing is wrong. Unfortunately, this can lead to incorrect conclusions from the "non-significant" findings.

## 9.8 The Mean Squared Error**

The mean squared error (MSE) is a measure of the expected squared difference between the estimator and the true parameter value:

$$MSE(\hat{b}) = E[(\hat{b} - b)^2]$$

To see how this decomposes into variance and bias, add and subtract $E[\hat{b}]$ inside the squared term:

$$MSE(\hat{b}) = E[(\hat{b} - E[\hat{b}] + E[\hat{b}] - b)^2]$$

Expanding the square:

$$MSE(\hat{b}) = E[(\hat{b} - E[\hat{b}])^2 + 2(\hat{b} - E[\hat{b}])(E[\hat{b}] - b) + (E[\hat{b}] - b)^2]$$

Taking expectations and noting that $E[\hat{b} - E[\hat{b}]] = 0$, the middle term vanishes:

$$MSE(\hat{b}) = E[(\hat{b} - E[\hat{b}])^2] + (E[\hat{b}] - b)^2$$

Which simplifies to:

$$MSE(\hat{b}) = var(\hat{b}) + bias(\hat{b})^2$$

We can **always** evaluate and compare linear models using the MSE or something akin to the MSE. The MSE is a function of both the variance of the estimator and the bias of an estimator.

## 9.9 The Ridge and Lasso Estimators

In statistics, it is common to hear about the so-called bias-variance tradeoff. Maybe we prefer a somewhat biased model, if it is estimated with more precision, or perhaps we prefer an unbiased model, even if it is estimated with less precision.

While the OLS model is BLUE in the presence of multicollinearity, severe multicollinearity (perhaps discovered by a large VIF or set of VIFs), can have adverse consequences. Recall that it increases the probability of Type II error, by way of large standard errors and small test-statistics; we may observe a very large $R^2$, yet no statistically significant effects; and/or some coefficients may be large and signed in the wrong direction – due to a large amount of error, of course.

Let's apply an alternative method then for model selection, what's known as **regularization** or **shrinkage**. The logic of these models are relatively simple:

Penalize regression parameters that are large. Why would we want to penalize a model with large parameters?

One of the consequences of multicollinearity is that it will lead to large coefficients. So, lets penalize large coefficients.

### 9.9.1 Ridge Estimator

The ridge estimator discussed extensively in Hastie et al. [2013, p. 63]. The idea is straightforward:

$$b_{Ridge} = min[(\sum_{i=1}^{N}(y_i - b_0 + \sum_J b_j x_{ji})^2 + \lambda \sum_J b_j^2] \tag{9.1}$$

The part on the left is well-known. We're just calculating the sum of squared residuals Yet, instead of just minimizing the SSR, we're minimizing $SSR + \lambda \sum_J b_j^2$. The $\lambda$ parameter penalizes the $b_j$ parameters. It's a value that is greater than or equal to zero. If it is zero, the model reduces to OLS. If it is non-zero, there is some degree of "shrinkage." The influence of the parameter (on the left), is canceled by the term on the right [Hastie et al., 2013, p. 64],

$$b_{Ridge} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \tag{9.2}$$

As we discussed then, we're adding a constant to the diagonals of the $\mathbf{X}^T\mathbf{X}$ matrix. The degree of bias is determined by the size of $\lambda$, and a consequence of this is that we also decrease the variance of the estimate. The ridge regressor will be biased, but it may have smaller variance (relative to OLS).

How do we choose $\lambda$? This uses 10-fold cross validation. So, split the data into 10 samples. One sample is held to be the testing sample. We fit the model on the $k - 1 = 9$ remaining samples, and calculate the Root Mean Squared Error on the testing sample. We then average. We do this for a vector of potential $\lambda$ values. We plot the results and choose the value of $\lambda$ with the smallest RMSE.

Let's apply Ridge regression to our Arizona precinct data, modeling Trump's vote margin as a function of median household income, percent Latino population, median age, and the Gini index. The plot shows the cross-validation results, with the optimal $\lambda$ indicated. We can see that there is too much shrinkage near the right of the figure and the RMSE is too large. However, the lowest RMSE is not when $\lambda = 0$. Note the difference between the Ridge coefficients and the OLS estimates!

```
library(glmnet)
```

```r
# Load the Arizona precinct data
load("precinct_voter_summary.rda")
load("precinct_tract_data.rda")

# Create percentage variables for demographics
precinct_voter_summary <- precinct_voter_summary |>
    dplyr::mutate(
        pct_latino = (tract_acs_latino / tract_acs_total_population) * 100,
        pct_white = (tract_acs_non_latino_white / tract_acs_total_population) * 100
    )

# Prepare data for Ridge regression
# Note: pct_democrat and pct_independent already exist in the dataset
az_complete <- precinct_voter_summary |>
    dplyr::select(
        trump_harris_margin, tract_acs_median_household_income,
        pct_latino, tract_acs_median_age, tract_acs_gini_index,
        pct_democrat, pct_independent
    ) |>
    na.omit()

# Create design matrix (predictors)
X_mat <- model.matrix(
    ~ tract_acs_median_household_income + pct_latino +
        tract_acs_median_age + tract_acs_gini_index +
        pct_democrat + pct_independent,
    data = az_complete
)[, -1]

# Response variable
y_ridge <- az_complete$trump_harris_margin

# 10-fold cross validation for Ridge (alpha = 0)
cv_ridge <- cv.glmnet(X_mat, y_ridge, alpha = 0, nfolds = 10)
plot(cv_ridge)
title("Ridge Regression: 10-Fold CV (Arizona Precinct Data)", line = 3)
```

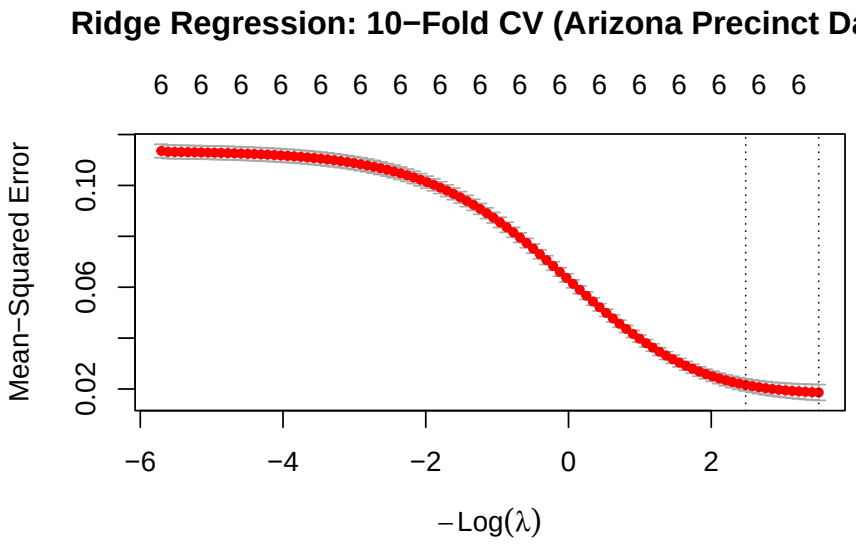



Figure 9.4: 10-fold cross validation for Ridge regression: selecting the optimal lambda

### 9.9.2 Choosing Lambda: Min vs 1-SE Rule

The cross-validation plot shows two important values, marked by vertical dashed lines:

- **lambda.min**: The value of $\lambda$ that minimizes the cross-validation error (lowest RMSE). This gives the best predictive performance on the training data.
- **lambda.1se**: A larger $\lambda$ value selected using the "1 standard error rule." This chooses the most regularized (simplest) model whose error is within one standard error of the minimum.

**Why use lambda.1se instead of lambda.min?** The 1-SE rule reflects a preference for parsimony and guards against overfitting. Models with slightly higher $\lambda$ are more stable – they have greater bias but lower variance. If two models have statistically indistinguishable performance (within one SE), we prefer the simpler one with more shrinkage. This is especially valuable when:

1. Your primary goal is **prediction on new data**, not just fitting the training data
2. You want **more interpretable** models with coefficients closer to zero
3. You're concerned about **overfitting** in small samples

```
Lambda (min RMSE): 0.03001082
```

```
Lambda (1 SE rule): 0.08350689

Ridge vs OLS Coefficient Comparison:
```

| Variable | Ridge ( .min) | OLS | Difference |
|---|---|---|---|
| (Intercept) | 0.9663 | 1.0710 | -0.1046 |
| Median Household Income | 0.0000001 | -0.0000000 | 0.0000001 |
| % Latino | 0.00094 | 0.00140 | -0.00046 |
| Median Age | 0.0009 | 0.0001 | 0.0009 |
| Gini Index | -0.2734 | -0.1883 | -0.0851 |
| % Democrat | -0.02085 | -0.02330 | 0.00246 |
| % Independent | -0.00973 | -0.01088 | 0.00115 |

Notice the difference relative to the OLS estimates! The ridge coefficients are shrunk toward zero. The degree of shrinkage depends on $\lambda$ – a larger penalty means more shrinkage, more bias, but also more stability.

### 9.9.3 The Lasso Model

The ridge estimator does not shrink a parameter to zero. We may wish to know if a variable has no effect – the ridge estimator only "proportionately" shrinks the parameters [Hastie et al., 2013, p. 69]. It's thus not quite as useful in terms of model specification – i.e., allowing the data to determine the model (and its parameters). The lasso regression, originally proposed by Tibshirani [1996], takes a different approach [Hastie et al., 2013, p. 72]:

$$b_{Lasso} = min[(\sum_{i=1}^{N}(y_i - b_0 + \sum_{J} b_j x_{ji})^2 + \lambda \sum_{J} |b_j|] \tag{9.3}$$

What this means is that, conditional on $\lambda$ this form of regularized regression will shrink the parameters to zero. The Lasso estimator is also quite easy to estimate in `glmnet`. Simply change $\alpha = 1$.

```
# 10-fold cross validation for Lasso (alpha = 1)
cv_lasso <- cv.glmnet(X_mat, y_ridge, alpha = 1, nfolds = 10)
plot(cv_lasso)
title("Lasso Regression: 10-Fold CV (Arizona Precinct Data)", line = 3)
```

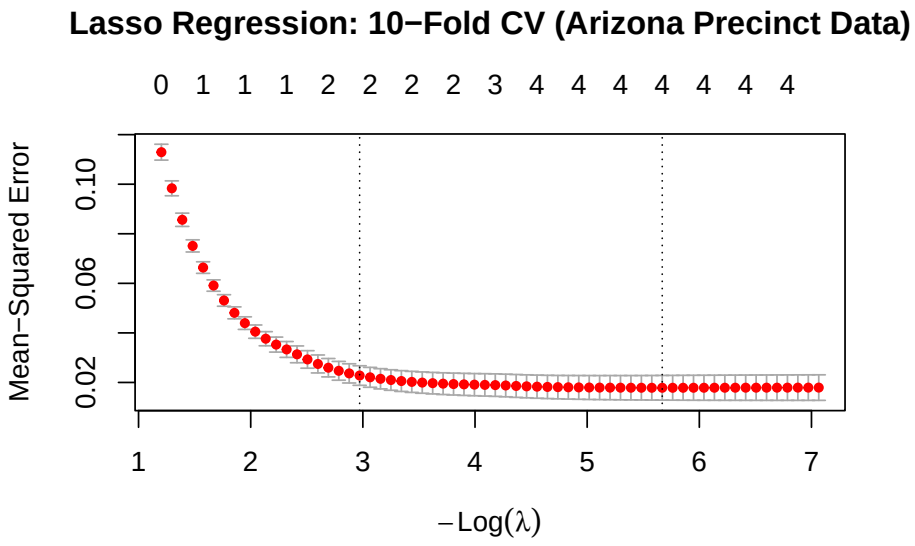



Figure 9.5: 10-fold cross validation for Lasso regression

`Lasso vs OLS Coefficient Comparison:`

| Variable | Lasso ( .min) | Lasso ( .1se) | OLS |
|---|---|---|---|
| (Intercept) | 1.0344 | 0.7215 | 1.0710 |
| Median Household Income | 0.0000000 | 0.0000000 | -0.0000000 |
| % Latino | 0.00108 | 0.00000 | 0.00140 |
| Median Age | 0.0000 | 0.0000 | 0.0001 |
| Gini Index | -0.1457 | 0.0000 | -0.1883 |
| % Democrat | -0.02302 | -0.01969 | -0.02330 |
| % Independent | -0.01014 | -0.00399 | -0.01088 |

### 9.9.4 The Bias-Variance Tradeoff in Regularization

We often encounter a tradeoff between bias and variance. The Ridge and Lasso estimators are **biased** models – they deliberately introduce bias to reduce variance. As $\lambda$ increases:

- **Bias increases**: coefficients are shrunk further from their true values
- **Variance decreases**: estimates become more stable across different samples

In contrast, OLS is unbiased but can have large variance when collinearity is severe. The key insight: **at some value of $\lambda$, the MSE of Ridge/Lasso can be less than the MSE of OLS**, even though Ridge/Lasso are biased.

This is precisely what cross-validation helps us find – the $\lambda$ that optimizes this tradeoff. When multicollinearity inflates OLS standard errors to the point where individual coefficients are unreliable, Ridge or Lasso regression offer an alternative (with some degree of bias).

**When to use Ridge/Lasso:** - When your goal is **prediction** rather than unbiased inference - When you face **severe multicollinearity** (VIF $>$ 10) - When you have **many predictors** relative to sample size - When you want to **regularize** model complexity

The following approach, however, is **not** a good solution to multicollinearity.

## 9.10 Dropping Relevant Variables

Often, the most appealing solution is to drop a variable. This may introduce a form of bias into the regression equation. For instance, assume the following **true** regression model:

$$Y_i = \beta_1 + \beta_2 X_1 + \beta_3 X_2 + \epsilon_i$$

But – due to collinearity in your data, you go ahead and estimate:

$$Y_i = \alpha_1 + \alpha_2 X_1 + \omega_i$$

$$\omega_i = \beta_3 X_2 + \epsilon_i$$

So you estimate $a_1$ and $a_2$.

The problem with this approach is that our estimate of $\alpha_2$ represents two things: the direct effect of $X_1$ but also the indirect effect of $X_1$ on $y$ through $X_2$

$$a_2 = \frac{\sum x_1 y_i}{\sum x_1^2}$$

$$a_1 = \bar{Y} - \alpha_2 \bar{X}_1$$

$$\beta_2 x_1 + \beta_3 x_2 + \epsilon_i - \bar{\epsilon_i}$$

In other words, we're just subtracting the expectation from the equation.

If we regress $X_2$ on $X_1$, then $X_2 = b_1 + b_2 X_1 + e_i$

$$a_2 = \beta_2 + \beta_3 \frac{\sum x_1 x_2}{\sum x_1^2}$$

$$b_2 = \frac{\sum x_1 x_2}{\sum x_1^2}$$

, so:

$$a_2 = \beta_2 + \beta_3 b_2$$

$a_2$ = direct effect of $X_1$ on Y, and the indirect effect of $X_1$ on $Y$ through $X_2$! It represents the unique impact of the covariate on the dependent variable *and* the effect shared with the excluded variable. The slope coefficient is biased – from how much $r_{x_1,x_2}$ are correlated.

Think of it this way. The true regression equation is:

$$Y = b_0 + b_1 X_1 + \gamma_1 X_2 + e$$

And lets say

$$X_2 = a_0 + a_1 X_1 + u$$

Write the "reduced form equation":

$$Y = b_0 + b_1 X_1 + \gamma_1 (a_0 + a_1 X_1 + U) + e$$

$$Y = b_0 + b_1 X_1 + \gamma_1 a_0 + \gamma_1 a_1 X_1 + \gamma_1 U + e$$

$$Y = \underbrace{b_0 + \gamma_1 a_0} + \underbrace{(b_1 + \ \gamma_1 a_1)} X_1 + \underbrace{(e + \gamma_1 U)}$$

If we exclude a relevant variable, the estimate is actually incorrect.

$$\mathbf{b} = (\mathbf{X^T X})^{-1} \mathbf{X^T y}$$

$$\mathbf{b} = (\mathbf{X^T X})^{-1} \mathbf{X^T} (\mathbf{X + Z + u})$$

$$E(\mathbf{b}) = \ + \underline{(\mathbf{X^T X})^{-1} \mathbf{X^T Z} + (\mathbf{X^T X})^{-1} \mathbf{U}}$$

The estimator is not unbiased – in fact, it's biased by however large the underlined component is.

So we can't really drop relevant variables in the presence of collinearity. This is a classic example of where the solution may introduce more problems than the original problem of collinearity itself. One more time just to be sure: **Don't drop a relevant variable because of collinearity**

## 9.11 A Summary

Table 9.4: Summary of Multicollinearity Consequences and Diagnostics

| Feature | With Multicollinearity |
|---|---|
| Bias of OLS | None – still unbiased |
| Variance of OLS | Inflated (larger SEs) |
| BLUE property | Still BLUE |
| Individual t-tests | Low power, often non-significant |
| Overall F-test | May still be significant |
| R-squared | Unaffected |
| Coefficient stability | Sensitive to small data changes |
| Primary diagnostic | VIF $> 4$ or sqrt(VIF) $> 2$ |

The worst mistake you can make is to drop theoritically important variables to "fix" a problem that is really just the data telling you it doesn't contain enough unique information to separate the effects. Report VIFs, discuss the implications, and resist the urge to let statistical convienence override substantive, theoretically informed reasoning (from scientific publications, theory).

# Bibliography

John Fox. *Applied Regression Analysis and Generalized Linear Models*. SAGE Publications, 3rd edition, 2015.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, 2nd edition, 2009. doi: 10.1007/978-0-387-84858-7.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, 2nd (corrected 12th printing) edition, 2013. doi: 10.1007/978-0-387-84858-7.

Paul R. Rosenbaum and Donald B. Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B*, 58(1):267–288, 1996.

Halbert White. A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica*, 48(4):817–838, 1980.